ON KNOWLEDGE REPRESENTATION AND DECISION MAKING UNDER UNCERTAINTY

Masoumeh Tabaeh Izadi

School of Computer Science
McGill University, Montréal

January 2007

A Thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfilment of the requirements for the degree of
Doctor of Philosophy

Your file Votre référence
ISBN: 978-0-494-32245-1
Our file Notre référence
ISBN: 978-0-494-32245-1

Dedicated to my loving family, who never stopped believing in me.

Acknowledgments

A journey is always easier when traveled with others. I have been accompanied and supported by many people throughout this work. I am pleased to have the opportunity to express my gratitude to all of them.

First and foremost, I would like to acknowledge a great debt of gratitude to my thesis advisor, professor Doina Precup whose inspiring and thoughtful guidance, and supervision made my thesis work possible. Discussions and regular meetings with Doina always helped shaping my thoughts and motivated me to work hard. During the past several years Doina has imparted innumerable lessons from practical instruction on teaching to research, writing, and presentation. I am honored to have had the opportunity to work with such a smart, knowledgeable, dedicated, and principle-centered person. Working with Doina was fruitful and enjoyable at the same time. Thank you does not seem sufficient but it is said with appreciation and respect.

I would also like to thank the members of my PhD committee who monitored my work and made efforts in reading my reports and providing me with valuable comments. I would like to thank professor Gregory Dudek for accepting to be in my committee despite his extremely busy schedule and for the extensive comments on my work; professor Monty Newborn who kept an eye on the progress of my work, was always available when I needed his advice, and has always built up my confidence; and professor Joelle Pineau whose expertise and constructive comments through discussions and meetings have had a direct impact on the final form and quality of this thesis.

I am grateful to my thesis external examiner, professor Brahim Chaib-draa, for generously spending time and energy in reading my thesis and providing valuable comments and constructive suggestions.

Within McGill, I have been fortunate to have been surrounded by gifted minds and accomplished people. I would like to thank professor Sue Whitesides the director of the school of computer science; professor Prakash Panangaden for giving many fascinating lectures to the lab and for accepting me to teach at McGill and thereby providing me with the incredible opportunity to experience exciting world of teaching; professor James Clark and professor Bettina Kemme for accepting to be in my thesis defense committee; and professor Brigitte Pientka for comprehensive leadership of women club in the department.

I wish to extend my warmest thanks to professor Peter Dayan for his support and supervision during a difficult time in my life and for giving me the opportunity to work at Gatsby Computational Neuroscience Unit at the University College of London where I had the chance to interact with top scientists including professor Zhoubin Ghahramani, professor David MacKay, Dr. Michael Duff, Dr. Fernando Pres Cruze and other colleagues for whom I have great regards.

My experience at McGill would certainly have less enjoyable without the fun time spent with my officemates. Robin Jaulmes, Mark Mouadeb, Swaminathan Mahadevan, Tomislav Vrjlicak, Norm Ferns, Pablo Castro, Robert Vincent, Amin Atrash, Melanie Coggan, Adrian Ghizaru and all the other members of Reasoning and Learning lab have been both friends and colleagues. It was very a rewarding experience to work with many collogues in particular, Erick Delage for proof reading my work, and always helping me with French translation; Philipp Keller, Marc Bellemare, Harry Seip, Zaid Zawaideh, Stephane Ross for inspiring discussions and for providing a lively atmosphere in the lab. I would like to express my appreciation to my collaborator Ajit Rajwade.

My thanks go to the staff in the school of computer science, whose efforts made my study in the Ph.D. program a pleasant process.

I feel fortunate to have had friends who stood by me through different stages of my work. I wish to thank Abeer Ghuneim, Danielle Azar, Malvika Rao, Miriam Zia for their continued moral support. I am indebted to Bohdana Ratitch who read many draft manuscripts, checked them and pointed out passages which I had not made clear. Thank you Bohdana for always make yourself available for to help me and for being a true friend.

I thank Mark Mercer for technical assistance, for walks around the campus and for making me feel okay about drinking so much tea and coffee.

I thank the people who deserve the most acknowledgement and to whom I dedicate this dissertation: my family. My family members have all instilled in me various lessons and tools to facilitate my academic success and I will be always grateful. I especially want to thank my parents, who taught me the value of hard work through their own example and whose love has been my guide through life; my sister Zarah for being an extraordinary source of advice, and for helping me realize the importance of research; my brother Majid for encouragements; my brother Saeed for being my eternal sunshine and my role model in determination, care, and self-accomplishments; and my sister Sara who had to put up with me for years while I was doing my Ph.D., for being a true friend and a compassionate assistant, and for constant care and support.

Abstract

Designing systems with the ability to make optimal decisions under uncertainty is one of the goals of artificial intelligence. However, in many applications the design of optimal planners is complicated due to imprecise inputs and uncertain outputs resulting from stochastic dynamics. Partially Observable Markov Decision Processes (POMDPs) provide a rich mathematical framework to model these kinds of problems. However, the high computational demand of solution methods for POMDPs is a drawback for applying them in practice.

In this thesis, we present a two-fold approach for improving the tractability of POMDP planning. First, we focus on designing good heuristics for POMDP approximation algorithms. We aim to scale up the efficiency of a class of POMDP approximations called point-based planning methods by designing a good planning space. We study the effect of three properties of reachable belief state points that may influence the performance of point-based approximation methods. Second, we investigate approaches to designing good controllers using an alternative representation of systems with partial observability called Predictive State Representation (PSR). This part of the thesis advocates the usefulness and practicality of PSRs in planning under uncertainty. We also attempt to move some useful characteristics of the PSR model, which has a predictive view of the world, to the POMDP model, which has a probabilistic view of the hidden states of the world. We propose a planning algorithm motivated by the connections between the two models.

Résumé

L'un des objectifs de l'intelligence artificielle est la misesur pied d'agents informatiques capable de prendre des décisions optimales dans leur environnement. Hors, plus souvent qu'autrement, la dynamique de cet environnement est cause d'ambiguitédans les observations de cet agent et d'incertitude dans la porteé de ces gestes. Une stratégie performante doit donc obligatoirement prendre en compte ces incertitudes. Les processus décisionnels de Markov partiellement observable (PD-MPO) fournissent un cadre mathématique riche pour modéliser ce genre de problémes. Cependant, la demande élevée de calcul des méthodes présentement disponibles pour résoudre des PD-MPOs requiert une grande quantité de calculs et limite l'application de ce modéle en pratique.

Dans cette thése, nous présentons deux approches pour rendre la la planification de PD-MPO plus traitable. D'abord, nous nous concentrerons sur la conception de bonnes méthodes heuristiques pour favoriser les algorithmes d'approximation de solution pour PD-MPO. Nous tenterons d'améliorer l'efficacité d'une classe de méthodes d'approximations de PD-MPO appelée planification par échantillonnages des états de croyance. Nous étudierons trois caractristiques des échantillons du groupe d'états de croyance qui peuvent influencer la performance de ces méthodes. Dans la deuxiéme section de cette thése, nous étudierons une représentation alternative pour la planification dans un environnement incertain appelée Reprsentation d'États Prédictifs (RÉP). Dans cette partie, nous décrirons l'efficacité de cette représentation. Nous tenterons également de transmettre certains avantages du modéle de RÉP, qui perçoit l'environnement á travers une série de prédictions, au modéle de PD-MPO, qui préfére poursuivre l'état caché de l'environnement. Finalement, nous proposerons un algorithmé de planification motivé par les affinités entre ces deux modéles.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

Decision making is ubiquitous in our day-to-day experiences. The human mind transforms a vast, complex array of incoming sensory stimuli into a sequence of purposeful actions. Although the neural mechanism of decision making is still unclear, one of the earliest goals of researchers in artificial intelligence has been to create autonomous systems that can perform similar decision making tasks. Clearly such systems need to maintain an internal model of the surrounding environment and of the interactions that are possible in it. Moreover, such systems must envision the effects of actions, and make predictions about the future. Hence, much of the research has focused on frameworks for modeling and control of dynamical systems. Dynamical system modeling is of great interest for a wide spectrum of disciplines including physics, chemistry, biochemistry, biology, economy, and sociology. A large number of applications involve taking a sequence of actions in which the effect of one action influences the expected utility of subsequent actions. The growing number of key applications of sequential decision making in dynamical systems has brought the development of computational approaches to this problem to the forefront of the research in several disciplines, including artificial intelligence. An intelligent decision making agent can be viewed as a black box which takes a temporal series of environmental signals and generates a series of actions. These two interconnected series constitute the observable experience of the agent. A standard mathematical framework used in operations research, control theory, and artificial intelligence to describe and solve

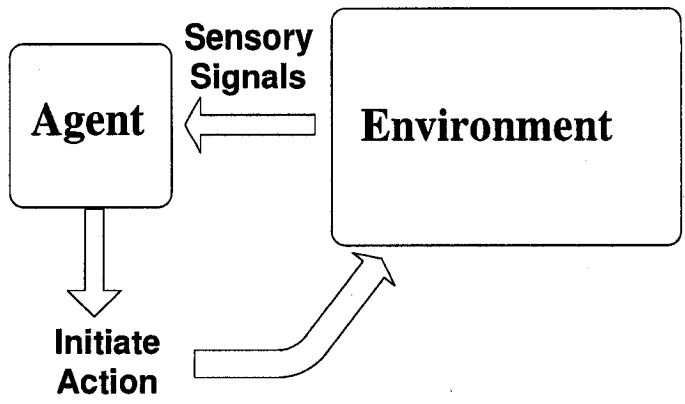


FIGURE 1.1. Graphical view of sequential decision making in dynamical systems

sequential decision problems is *Markov Decision Processes (MDPs)* [Puterman, 1994; Howard, 1960]. A decision maker, or *agent*, takes actions on the basis of a signal from the environment called the *state* of the environment. In an MDP environment, the sensory inputs completely identify the state of the environment. Moreover, the new state of the environment after making a decision and taking an action can be predicted (probabilistically) only based on the current state and action. In the MDP framework, the assumption is that the states are *Markovian*, which means the response of the environment at each point in time depends only on the state at the previous time and the action taken by the agent. Formally, an MDP is defined by a tuple $\langle S, A, T, R \rangle$, where S is the environment's state space, A is the agent's action space, T is a probability distribution describing the state transitions, called *transition function*, and R is a function that represents expected immediate rewards or utilities for different states and actions, called the *reward function*.

Ideally, we would like the state signals to inform the agent of everything about the environment, such that the agent can precisely predict the effects of its action, in terms of the immediate reward and the next state of the environment. However, agents act in real environments and usually do not have correct and complete information about the states of the world in which they exist. Therefore it is important for these agents to be able to choose their actions in the presence of uncertain information. The agent box must maintain its knowledge of the process, often in terms of some notion of internal state, in order to

analyze its experience with the system and decide how to act. MDPs do not typically model this situation.

A more general approach for tackling such problems is to define and solve a *Partially Observable Markov Decision Process* (POMDP). POMDPs maintain a probabilistic model that reflects the imperfect information about the current state of the environment and the effects of actions. The agent tries to construct a Markov state signal from a non-Markov state representation of the environment. Formally, a POMDP is an MDP in which the agent is unable to observe the current true state of the environment. A POMDP model is described by a tuple $\langle S, A, T, R, Z, O \rangle$, where S, A, T, and R describe a Markov decision process, Z is a set of observations the agent can experience in its world, and O is the probability distribution over possible observations, depending on the state and action, called the *observation function* (see Chapter 2 for more details). POMDPs were originally introduced in the control theory and operations research literature [Astrom, 1965; Cheng, 1988b; Drake, 1962; Lovejoy, 1991b; Monahan, 1982; Sondik, 1971; White, 1991]. They have been widely applied in modeling decision making under uncertainty in various domains such as artificial intelligence, medical decision making, mining engineering, robotics, etc. Prominent application examples include robot navigation [Montemerlo *et al.*, 2002; Koenig and Simmons, 1998; Cassandra *et al.*, 1996], spoken dialog management [Paek and Horvitz, 2000; Roy *et al.*, 2000; Zhang *et al.*, 2001; Williams *et al.*, 2005], task and resource allocation [Chong *et al.*, 2004; Tambe and the TEAMCORE group University of Southern California, 2005], designing automated personal assistants [Varakantham *et al.*, 2005], preference elicitation [Boutilier, 2002; Boutilier *et al.*, 2005; Braziunas, 2006], and othert areas [Cassandra, 1998b].

To *solve* a POMDP means to find an optimal choice of actions, also called a *policy*, so that the total expected utility over a given period of time is maximized. The great generality of the POMDP model implies that there is no single method for solving all POMDPs effectively. It is well known that exactly solving POMDPs is very costly in terms of computation. This is due to the fact that hidden state of the Markov model cannot be identified exactly based on the interactions with the environment. Therefore, the agent

cannot form an optimal policy based on the exact hidden state. Instead, it must keep a sufficient statistic of its complete history of actions and observations. This is known as the *curse of history* in the POMDP literature. The sufficient statistic in POMDPs is called the *belief state*, and consists of a probability distribution over all the unobservable states. If the environment is finite, the belief state is typically represented as a vector in which each element specifies the agent's current belief of being in a particular state of the environment. This immediately implies that the agent needs to consider a distribution of dimension equal to the number of underlying hidden states in order to reason about every possible belief. This problem is known as the *curse of dimensionality* in the POMDP literature. A huge research effort in the field has been devoted to tackle these two problems by using assumptions about the structure of the POMDP [Feng and Hansen, 2001; Guestrin *et al.*, 2001; Hansen and Zhou, 2003a], different representations [McCallum, 1993; Boutilier and Poole, 1996; Roy, 2003; Ng and Jordan, 2000; Hansen and Feng, 2000], and efficient approximate solution methods [Braziunas, 2003b; Yu and Bertsekas, 2004; Hauskrecht, 2000a; Li and Littman, 2005; Pineau *et al.*, 2003; Parr and Russell, 1995; Wang *et al.*, 2006] (also see excellent surveys by Murphy 2000 and Braziunas 2003a for more on POMDP approximation methods).

1.1. Thesis Objectives

In this thesis we investigate novel approaches that target the two curses of dimensionality and history in order to solve sequential decision making problems under uncertainty. Moving towards this goal, we study first the point based approximation framework, one of the best approximate planning methods for POMDPs. A simple yet important observation behind this class of POMDP approximation methods is that the continuous belief space of a POMDP is not necessarily what needs to be considered as planning space given a known initial state distribution. Instead, one should plan only for the belief states which can be experienced by the agent. In one part of this thesis, we investigate using several criteria to guide the search for selecting belief states in Point-Based Value Iteration (PBVI) [Pineau

et al., 2003]. One contribution of the thesis is to develop and test heuristics that allow tailoring the set of required points based on several properties.

The curse of dimensionality is countered by reformulating the sequential decision making problem using an alternative state representation, which can introduce a more compact model than POMDPs. This representation, called *Predictive State Representation (PSR)* [Littman *et al.*, 2001; Singh *et al.*, 2004] can provide dimensionality reduction in prediction and control problems. A number of representations for dynamical systems have been proposed during the past two decades [Jaeger, 2000; McCallum, 1995b; Rivest and Schapire, 1994; Dayan, 1993; Platzman, 1997]. However, they either impose restrictions to the underlying environment (i.e, they are not as general as POMDPs), or they do not seem to provide any advantages over POMDPs. PSRs [Littman *et al.*, 2001] seem appealing for two main reasons. First, PSRs are grounded in the sequence of actions and observations of the agent, and hence relate the state representation directly to the agent's experience. Second, PSRs offer a representation for dynamical systems which is as general as POMDPs, and can be potentially more compact than POMDPs. The fundamental characteristics of the model make it potentially easier for learning and planning. One of the goals of this thesis is to investigate how the PSR representation can be augmented and used in a planning framework and how it can affect the performance of planning techniques. Another goal of the thesis is to characterize features of the PSR representation which can facilitate decision making.

1.2. Contributions

1.2.1. Belief Selection for Point-Based Value Iteration

On one hand, the quality of the approximate solution found by point-based methods is improved as more belief points are included. On the other hand, the performance of these algorithms can be improved by eliminating unnecessary points and concentrating on more important belief points, in order to obtain a good approximation of the optimal solution. We present new heuristics that address the problem of quantifying how large the set of points

considered in these methods should be, in order to get a sufficiently good approximation in reasonably little time.

We propose three heuristics based on different metrics for measuring the similarity of beliefs from the reachable belief space. One of these methods is especially robust to parameter settings. Chapter 4 discusses these approaches and evaluates them empirically on several standard test problems.

1.2.2. Structure Exploration through State Representation

In a complex dynamical system, useful abstractions of knowledge can be essential to an autonomous agent for efficient decision making. This is the focus of the second part of the thesis.

We study structure that can be detected and exploited based on mathematical properties of predictive representations. We point out a special case in which a strict reduction in the number of states is obtained by a subclass of PSRs called *linear PSRs* [Izadi and Precup, 2005a]. We define a linear dependence property exhibited by states which makes it possible to reduce the number of states. However, this reduction ignores the state values. Therefore, if a linearly dependent state has a distinctive value then we lose information by this type of abstraction. Considering reward as part of observation in the PSR definition of a test introduces a similar effect in terms of the model minimization and respects value equivalence as well as equivalence in dynamics.

To make an analogy in the POMDP framework, we also incorporate reward values in the POMDP definition of belief states. Using rewards in the belief updates can help a POMDP agent to distinguish between belief states in some domains. We show that this is useful in solving POMDPs both exactly and approximately [Izadi and Precup, 2005b].

1.2.3. Using Predictive State Representation in Control Problems

Since the representational power of PSRs is equivalent to the belief state representation in POMDPs, one can imagine PSR planning algorithms working in the context of controlling dynamical systems. The third part of this thesis focuses on the applicability of

PSRs for control purposes, and introduces new algorithms for planning with this representation. This is done by incorporating PSRs into POMDP control algorithms. We developed a planning algorithm based on a lookahead search [Izadi and Precup, 2003]. Experimental results on standard domains confirm that the empirical performance of exact PSR planning is similar to belief-based planning except for highly structured cases in which PSRs provide considerable compression in the state space.

We investigate the possibility of developing point-based planning methods for PSRs. We present an algorithm for approximate planning in PSRs, based on an approach similar to point-based value iteration (unpublished). The point-based approach turns out to be a good match for the PSR state representation. We present empirical results on several standard POMDP domains which confirm that PSRs provide an advantage compared to belief states. The power of our approach is more pronounced in domains with special structure.

1.3. Statement of Originality

Portions of this thesis have previously been published in peer-reviewed conference proceedings [Izadi and Precup, 2003; 2005a; 2005b; Izadi *et al.*, 2006; 2005], and presented at workshops [Izadi and Precup, 2006]. The material presented in this thesis contains more extensive discussions of the proposed techniques and problems investigated compared to the content of the published papers. The thesis also contains a significantly more extensive literature review and in-depth discussions of the relationships between the proposed methods and existing approaches. The experimental studies presented in Chapters 4, 5, and 6 contain additional results and different evaluations not included in [Izadi and Precup, 2003; 2005a; 2005b; Izadi *et al.*, 2006; 2005; Izadi and Precup, 2006].

1.4. Thesis Outline

The rest of the thesis is structured as follows.

Chapter 2 illustrates the basic POMDP framework, discusses methods for solving POMDPs, problems with POMDP exact solution methods, and the state of the art in POMDP approximate planning techniques.

Chapter 3 describes the predictive state representation framework (PSR), and introducing rewards into this representation. In this chapter we also look at different definitions of tests and their potential effects on planning.

Chapter 4 considers the general problem of POMDP approximation methods. We propose new heuristics for belief selection in the point-based value iteration algorithm. These methods include a technique inspired by PSRs for seeking good points, a value-based approach to point selection, and reachability analysis and its implications on the PBVI algorithm. Experimental results using these new methods are also presented.

Chapter 5 defines special structure that can be captured by linear PSRs and analyzes this structure compared to other properties previously defined in the literature for state abstraction in dynamical systems. We also propose reward-based belief updates in POMDPs to reduce the uncertainty about the states. Our empirical results show that considering rewards in belief updates is advantageous for planning.

Chapter 6 describes our approach for solving sequential decision problems in partially observable systems which are modeled by PSRs. We propose new planning methods for PSRs based on a forward search technique and based on point-based approximation methods. We evaluate these methods on several standard domains and obtain comparable results with the corresponding methods to solve POMDPs. We highlight, both in theory and in experiments, a special case of dynamical systems in which point-based approximation using PSRs is superior to the same approximation using POMDPs.

Chapter 7 includes a concluding summary and possible future extensions of this work.

CHAPTER 2

Partially Observable Markov Decision Processes

Chapter Outline

In this chapter, we provide background information on Partially Observable Markov Decision Processes (POMDPs), a formalism for modeling sequential decision making problems in real world systems. We introduce the POMDP model description in Section 1. In Section 2 we describe what it means to solve a POMDP problem and discuss the exact POMDP solution methods. In Section 3 we review the main approaches for approximate planning in POMDPs and the state-of-the-art algorithms utilizing these approaches. A summary and concluding remarks are given in Section 4.

Sequential decision making problems in dynamic environments can be modeled as Markov decision processes (MDPs). In an MDP environment, the new state of the environment after making a decision and taking an action can be predicted using only the current state. But making decisions in many applications involves dealing with uncertain information about the underlying process. In other words, the states of the environment are not completely observable. Instead, only some noisy sensations of the states are available to the agent. The uncertainty can also be due to partial specification of the transitions between states of the system (i.e., action effects). An intelligent decision making agent is required to have knowledge or beliefs about its environment and its dynamics, and must sequentially make decisions on a course of action which maximizes the agent's expected utility.

POMDPs can model this kind of decision problem by maintaining a probabilistic model that reflects the imperfect information about the current state of the environment and the effects of actions.

POMDP methods try to construct a Markov state signal from a non-Markov state representation of the environment. To effectively make decisions in a partially observable world, it is necessary to use some form of memory of the past actions and observation signals to help in the disambiguation of the states of the world, and the POMDP framework provides such an approach. As a general framework with many application domains, POMDPs have drawn increasing attention and a huge research effort has been devoted to this field.

In this chapter, we briefly present notation, concepts, algorithms and results that enable us to define the problems addressed in this work. For further details on POMDPs and their solution methods we refer interested readers to [Kaelbling *et al.*, 1998; Lovejoy, 1991b; Littman *et al.*, 1995b; Cassandra, 1998a; McCallum, 1995b; Boutilier and Poole, 1996].

2.1. Model Description

Formally, a POMDP is defined as a generalization of an MDP and consists of:

- *State space S*: The world is modelled by a set of distinct states S which can be finite, countably infinite, or continuous in general. Throughout the thesis we assume S is finite with $|S| = n$. The state at time step t is denoted by s_t. In a POMDP, these states are not directly observed by an agent and only a probability distribution over these states can be maintained by the agent.
- *Action space A*: The states of the world can be influenced by the agent by taking actions. The set of available actions can be discrete,countably infinite or continuous, but here we only consider a finite set of actions A and denote by a_t the action choice of an agent at time t. The behavior of the agent in selecting actions is determined by a policy (which will be discussed later on in this chapter).
- *Observation space Z*: The observable feedback from the world to the agent consist of a set of observations. We assume a finite set of possible observations $Z = \{z_1, z_2, ..., z_k\}$ and we denote by z_t the observation made by the agent at time

t. In a POMDP, state aliasing can occur in the sense that an agent can make the same observations at different states of the world. In the fully observable setting (MDP) the set of observations is the same as the set of states, and at each time step $s_t = z_t$.

- *Transition function* $T : S \times A \times S \rightarrow [0,1]$: The actions of the agent can influence the states of the world. The effect of actions is captured by the transition function T. Intuitively, $T(s,a)$ defines a distribution over next states when an action a is taken in state s. Formally,

$$T(s,a,s') = P(s_{t+1} = s' | a_t = a, s_t = s) \tag{2.1}$$

  The transition function exhibits the Markov property, i.e., the probability of transition to state s_{t+1} depends only on the current state s_t, and the current action a_t and does not depend on the previous states and actions.

- *Observation function* $O : A \times S \times Z \rightarrow [0,1]$: The uncertainty about the exact state of the world is reflected by observation function. $O(a,s',z)$ is a probability distribution over the observation space when the agent takes action a and makes a transition to state s':

$$O(a,s',z) = P(z_{t+1} = z | a_t = a, s_{t+1} = s') \tag{2.2}$$

- *Initial belief state* b_0: A probability distribution over unobservable MDP states, where $b_0(s) = Pr(s_0 = s)$
- *Reward function* $R : S \times A \rightarrow \Re$: Conceptually, the world issues rewards after each state transition. The expected immediate reward gained by the agent for taking each action in each state is defined by the reward function. The real value $R(s,a)$ denotes the expected immediate reward gained by taking action a in state s.
- *Discount factor* γ: The importance of the rewards achieved by the agent can be weighted by a discount factor. Typically the more delayed the reward is, the

less important its value becomes. The discount factor of a POMDP is a constant $0 < \gamma \leq 1$.

Generalizations to the case of infinite states, actions and observations [Williams *et al.*, 2005; Spaan, 2006; Hoey and Poupart, 2005; Porta *et al.*, 2005] are possible for many algorithms, but it generally makes the problems more complex and harder to solve. The POMDP formulation considered throughout this thesis assumes that all model components are finite and provided in the problem definition.

2.2. POMDP Belief State

In a POMDP setting, we still maintain the Markov assumption: future states and observations are conditionally independent from past states and observations given knowledge of the current state. However, in contrast to the MDP, the current state is not accessible; only the actions and observations are known. Instead, the state is estimated using a generic internal state of a POMDP agent called *belief state*. [Astrom, 1965] first described belief states, and most of the existing POMDP solution methods are based on belief state estimation. Belief states are sometimes referred to as *information states* [Cassandra, 1998a].

At each time step t, the agent maintains an estimation of the state of the world, $b_t = (P(s_1^t)...P(s_{|S|}^t))$, which is a posterior distribution over the state space at time t:

$$P(s_t = s|z_t, a_{t-1}, z_{t-1}, ..., a_0, b_0) \; \forall s \in S \tag{2.3}$$

where b_0 is the initial distribution over states. Maintaining the entire history of the agent can become computationally difficult as the time increases. The belief state is a sufficient statistic for the sequence of actions observations in the past. Therefore, the agent does not need to remember the entire history if it has access to b_{t-1} [Smallwood and Sondik, 1973]. This implies a new form of the Markov assumption: future observations and beliefs are conditionally independent from the past, given the current belief.

The transition function of this MDP is:

$$P(b_{t+1}|b_t, a_t) = \sum_z P(b_{t+1}|a_t, b_t, z) P(z|a_t, b_t) = \sum_z P(b_{t+1}|a_t, b_t, z) \sum_s b_t(s) P(z|a_t, s) \tag{2.4}$$

Note that exactly one belief point will have probability 1 at time step $t+1$ after the observing z_{t+1}. Figure 2.1 describes this graphical model.

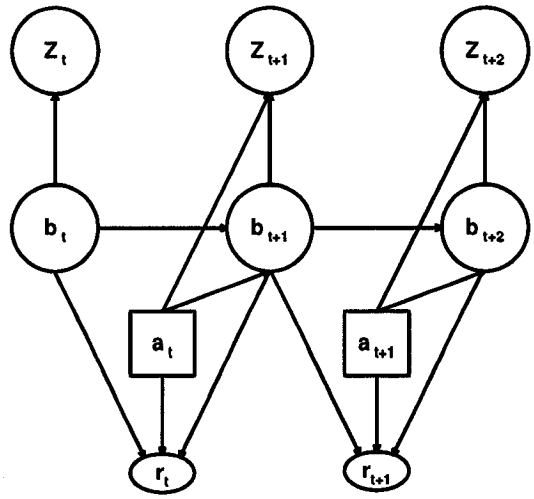


FIGURE 2.1. An illustration of system dynamics by POMDP model.

The belief at time $t+1$ can be computed using Bayes rule. Each component of b_{t+1} corresponding to state s_i can be determined as follows:

$$b_{t+1}(s_i) = \frac{P(z_{t+1}|s_{t+1}=s_i,a_t,b_t)P(s_{t+1}=s_i|a_t,b_t)}{P(z_{t+1}|a_t,b_t)} = \frac{O(a_t,s_i,z_{t+1})\sum_s T(s,a_t,s_i)b_t(s)}{\sum_{s'} O(a_t,s',z_{t+1})\sum_s T(s,a_t,s')b_t(s)} \tag{2.5}$$

A POMDP is a belief state MDP: that is, an MDP with a state space that consists of beliefs. Figure 2.2 depicts the continuous belief space of a POMDP with three states in the underlying MDP. Note that although the belief space is continuous, if an initial belief is fixed, only a countable number of beliefs can be reached. In a finite amount of time, the number of reachable beliefs becomes finite.

2.3. Solving POMDPs

The goal of a POMDP agent is to find a long term plan or *policy* for acting in such a way as to maximize the total expected reward received. The best such plan is called an *optimal policy* or an *optimal solution* for the POMDP. The belief state is known to be a sufficient statistic for computing an optimal policy in POMDPs [Astrom, 1965]. Hence a policy is a mapping $\pi : B \rightarrow A$.

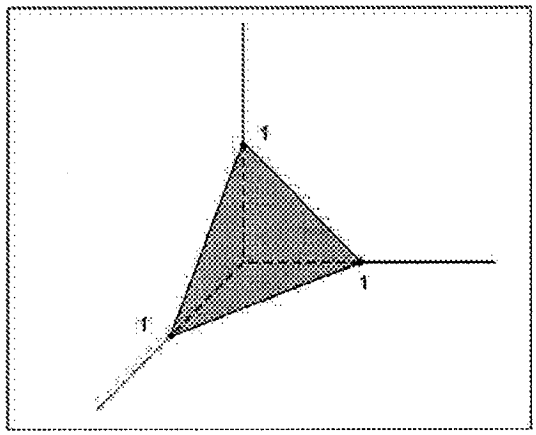


FIGURE 2.2. Belief space of a POMDP with three underlying states.

Given a policy π, it is necessary to know how good it can be when executed. The evaluation of a policy π is based on the rewards that an agent expects to receive following π in the future h steps. We call h the *horizon* of interest. One possible way of evaluating a policy is to look at $E[\sum_{t=1}^{h-1} r_t]$, where r_t is the reward received at time step t. Sometimes this is not quite appropriate since the time of receiving a reward is also important to the agent. Earlier rewards are more valuable to the agent. Hence, rewards are discounted in this case and the agent's total expected reward in h time steps is measured by $E[\sum_{t=0}^{h-1} \gamma^t r_{t+1}]$ where $(0 < \gamma \leq 1)$ is the discount factor defined by the POMDP model.

The amount of total expected reward that an agent can accumulate over its lifetime as given by the horizon h and following a policy π is called the *value function* of π. Most of the POMDP algorithms are based on estimating a value function. A value function V^π of the policy π defines the value for each belief state under policy π.

$$V^\pi : \mathcal{B} \rightarrow \Re \tag{2.6}$$

The value function assigns to each belief state b the expected value of the total reward the agent can get in the future, given that its starting point is b. The optimal policy π^* in particular is the one that maximizes the total expected future reward:

$$\pi^*(b) = \arg\ \max_\pi E[\sum_{t=0}^{h-1} \gamma^t r_{t+1} | b] \tag{2.7}$$

Finding good policies for POMDPs is generally difficult. POMDPs with deterministic transition and observation functions (*deterministic POMDPs*) and finite horizon are known to be NP-complete, while *stochastic POMDPs* are PSPACE-hard for finite horizon. A number of complexity bounds for POMDPs are discussed in [Mundhenk *et al.*, 1997; Madani *et al.*, 2003; Littman, 1997].

To get an intuition of why POMDPs are very hard to solve, suppose the agent is allowed to make h decisions. If there are n actions and m observations available in the model, then there are $(mn)^h$ possible action-observation sequences. Each such sequence can be considered as a policy, and could be an optimal policy for a particular subset of the belief space and consequently, useful in finding an optimal policy over the continuous belief space. This phenomenon, which is called the *curse of history*, is one reason for the difficulty of solving POMDPs. Another reason is that in a POMDP with n underlying states we need to find and represent the value function (or policy) over a continuous $(n-1)$-dimensional space of beliefs. This problem is called the *curse of dimensionality*[Pineau, 2004]. A POMDP with a small number of underlying states could have a very complex representation for its value function. Conversely a large POMDP can have a simple value function. Therefore, the curse of dimensionality and the curse of history can potentially influence independently the complexity of solving POMDPs.

2.3.1. Representing the POMDP Value Function

Maintaining and updating independent values for infinitely many belief states is infeasible. However, it has been shown for a finite horizon that the value function over all belief states can be represented exactly by a convex piecewise linear function [Smallwood and Sondik, 1973].

The optimal value function for a POMDP is defined as:

$$V^*(b) = \max_{a\in A} \sum_{s\in S} R(s,a)b(s) + \gamma \sum_{z\in Z} P(b'|b,a,z)P(z|a,b)V^*(b') \qquad (2.8)$$

where $\sum_{s\in S} R(s,a)b(s)$ is the immediate reward at belief state b, and b' is the next belief state which is determined by the current belief state b, action a and observation z as a vector

of probabilities (equation (2.4)). The above equation is known as the *Bellman optimality equation* for POMDPs. [Sondik, 1971] showed that the value function at any finite horizon h can be represented by a set of vectors: $\Gamma_h = \{\alpha_1, \alpha_2, ...\alpha_k\}$ such that each vector represents an $|S|$-dimensional hyperplane. Each α-vector defines the value function over some region of the belief space. Therefore the above equation for a value function of horizon t can be rewritten as:

$$V_h(b) = \max_{\alpha \in \Gamma_h} \sum_{s \in S} \alpha(s)b(s) \tag{2.9}$$

Associated with each policy π is a set of *policy trees* of the form π_p and each such tree has an α-vector α_p. A policy tree π_p is a fixed policy shown as a tree of actions conditioned on observations. The length of the policy tree is the horizon of interest. Figure 2.3 shows

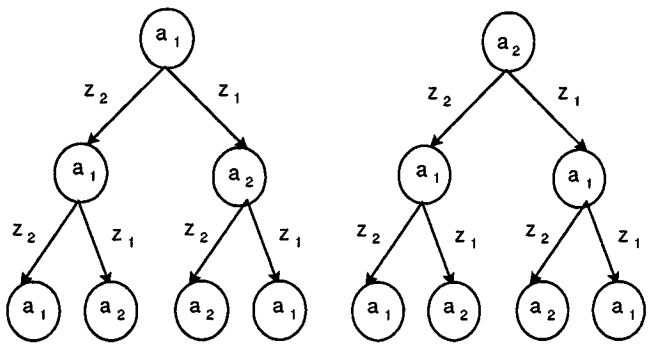


FIGURE 2.3. A sample of two 3-step policy trees for a problem with two actions and two observations

a policy tree of length 3. An α-vector represents the value V_{π_p} of taking actions from each state s_i based on a particular policy tree. Therefore, V_{π_p} is a vector of length $|S|$. The value of executing a policy tree π_p from a belief state b is :

$$V_{\pi_p}(b) = \sum_{s \in S} V_{\pi_p}(s)b(s) \tag{2.10}$$

Therefore, in terms of the α-vectors we can write:

$$V_{\pi_p}(b) = \alpha_p.b$$

Each α-vector α_p defines a hyperplane in belief space. Note that in general, each hyperplane is associated with the action at the root of its policy tree. The above equation means

that the value function is a linear function of the belief distribution. The optimal value of a belief state b in horizon h is the maximum over all hyperplanes associated with policy trees π_p of depth h:

$$\begin{aligned} V_h(b) &= \max_p \alpha_p . b \\ &= \max_p \sum_{s \in S} R(s,a)b(s) + \gamma \sum_{s' \in S} \left(\sum_{s \in S} P(s'|s,a_p)b(s) \right) \sum_{z \in Z} P(z|s',a_p) V_{\pi_{pz}}(s') \end{aligned}$$

Applying the *max* operator indicates that $V_h(b)$ is the upper surface of all the linear surfaces generated by all possible policy trees. Therefore, $V_h(b)$ is a piecewise linear and convex function. The optimal value function only contains the hyperplanes corresponding to optimal actions. Hence, not all hyperplanes are necessary or useful to represent the value function. To be useful for a value function representation, a hyperplane α must be the maximum for some b (Figure 2.4). If Γ contains only useful hyperplanes then it is called a *parsimonious set* [Littman *et al.*, 1995a]. POMDPs with as few as two states can have an optimal value function which needs an exponential number of α-vectors in their parsimonious set.

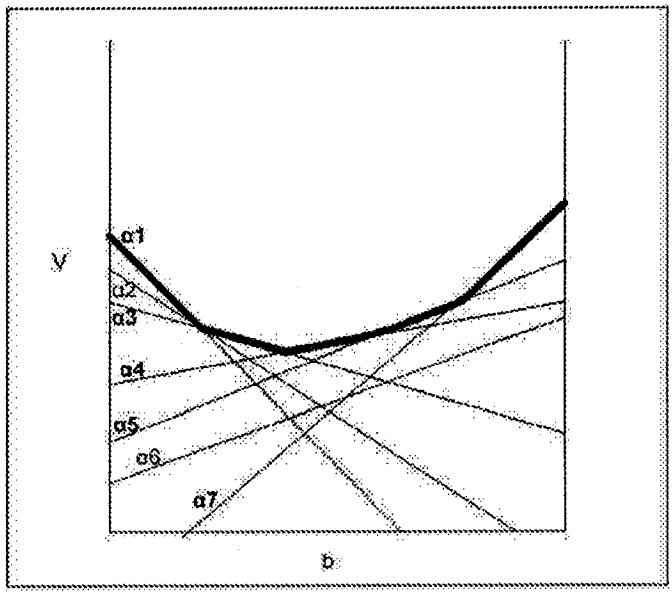


FIGURE 2.4. An illustration of a value function as the superposition of hyperplanes (line-segments here in 2D). Vectors $\alpha_1, \alpha_3, \alpha_4, \alpha_5$ and α_7 are useful in constructing the value function.

The optimal policy can be computed based on the optimal value function:

$$\pi^*(b) = \arg\max_{a \in A} \sum_{s \in S} R(s,a)b(s) + \gamma \sum_{z \in Z} P(b'|b,a,z)P(z|a,b)V^*(b') \tag{2.11}$$

A globally optimal policy is known to exist for MDPs when $\gamma < 1$ [Howard, 1960]. However, even finite horizon POMDPs are undecidable [Madani *et al.*, 2003].

2.3.2. Value Iteration Methods

Many POMDP planning methods work by constructing a finite representation of a value function over the continuous belief space and then iteratively updating this representation by expanding the horizon of the corresponding policy until a desired depth is reached or the value function converges to a stable value. This technique is called *value iteration* [Bellman, 1957; Bertsekas, 1987]. The basic idea behind the value iteration algorithm is to construct policies gradually, one horizon at a time, and to reuse the policy found for the preceding horizon. The algorithm proceeds by using a value update on all α-vectors.

Algorithm 1 Exact Value Iteration Algorithm

INPUT: POMDP model and horizon h
Initialize α_0 to a vector of all zeroes
$\Gamma = \{\alpha_0\}$
for all $a \in A$ **do**
  $\alpha^{a,*}(s) = R(s,a)$
  $\Gamma^{a,*} = \{\alpha^{a,*}(s_0), \ldots, \alpha^{a,*}(s_n)\}$
end for
for all $t = 1$ to $t = h$ **do**
  for all $a \in A$ **do**
    for all $z \in Z$ **do**
      for all $\alpha' \in \Gamma$ **do**
        $\alpha^{a,z}(s) = \gamma \sum_{s' \in S} T(s,a,s')O(s',a,z)\alpha'(s')$
        $\Gamma_t^{a,z} = \Gamma_t^{a,z} \cup \{\alpha^{a,z}\}$
      end for
    end for
    $\Gamma_t^a = \Gamma^{a,*} \oplus \Gamma_t^{a,z_1} \oplus \Gamma_t^{a,z_2} \oplus \ldots \oplus \Gamma_t^{a,z_{|Z|}}$
  end for
  $\Gamma_t = \cup_{a \in A} \Gamma_t^a$
  $\Gamma = \Gamma_t$
end for
Return Γ

Dynamic programming is used to update these vectors. Such updates generate new vectors to be added to Γ.

Different exact POMDP algorithms have different approaches to this problem. A common update implementation solves a set of linear programs. Once a single round of value iteration has been performed Γ is examined to remove useless dominated vectors and create a parsimonious set ready for the next round. This process, which is known as *pruning*, can be complex since Γ contains exponentially many vectors.

2.3.2.1. Sondik/ Monahan's Algorithm

The simplest exact POMDP method is Sondik / Monahan's enumeration algorithm [Sondik, 1971; Monahan, 1982], which implements the exact value iteration algorithm (2.3.2) and can find an optimal policy for a specified horizon. This algorithm performs dynamic programming to compute increasingly more accurate values for each belief state b. The initial value function is:

$$V_0(b) = \max_{a \in A} \sum_{s \in S} b(s)R(s,a) \tag{2.12}$$

The value function at the horizon of $t+1$ is constructed from the value function at the horizon t recursively:

$$V_{t+1}(b) = \max_{a \in A} \sum_{s \in S} b(s)R(s,a) + \gamma \sum_{b'} P(b'|b,a)V_t(b') \tag{2.13}$$

where b' is the next belief state given the current belief b and the action a.

The representation of the corresponding value function, Γ_{t+1}, can be generated from the set of α-vectors Γ_t using the operations known as *exact value backup*. First, we generate an intermediate sets of α-vectors for each action-observation pair and then we take the cross sum [1] over all observations to create a single hyperplane for each action. The union of all such vectors represents the set Γ_{t+1}.

The algorithm enumerates all hyperplanes in Γ_t and performs a dynamic programming update on each one. This method results in a large number of new vectors and is intractable

[1] $\oplus$ denotes the cross-sum operator, $p \oplus q = [p_1 + q_1, \ldots, p_n + q_n]$

because of the exponentially growing number of conditional plans. In the worst case the new value function needs $|\Gamma_{t+1}| = O(|A||\Gamma_t|^{|Z|})$ α-vectors. The Linear Support algorithm of [Cheng, 1988a] is also very similar.

2.3.2.2. Witness Algorithm

Similar to the definition of a value function is the one of the *action-value function*, which corresponds to the expected reward for choosing a fixed action a in the first time step and acting optimally afterwards.

$$Q^*(b,a) = \sum_{s \in S} R(s,a)b(s) + \gamma \sum_{z \in Z} P(b'|b,a,z)P(z|b,a)V^*(b') \tag{2.14}$$

Using this definition, equation (2.8) can be rewritten as:

$$V^*(b) = \max_{a \in A} Q^*(b,a) \tag{2.15}$$

In a similar manner to V, Q is represented by a set of piecewise linear and convex functions U^a. The *Witness* algorithm [Littman, 1994; Cassandra *et al.*, 1994], is similar to Sondik's algorithm in that it looks for a point in where the current set of α-vectors are not dominant. However, instead of computing value function V_t for all actions and all observations at the same time, the witness algorithm considers computing sets of vectors for each action individually and then focusing on each observation one at a time. Therefore, it constructs Q_t^a of t-step policy trees for each action a and finds the best value function for each action then computes V_t by taking the union of the Q_t^a for all actions and eliminating dominated vectors. The basic goal of the witness algorithm is to find a parsimonious representation of U^a for each action a in a dynamic programming fashion.

In each iteration of dynamic programming updates for the set U^a, the algorithm looks for a belief state b which is an evidence or witness to the fact that the current vectors in U^a are not yet a perfect representation of $Q_t^a(b)$ and have to be expanded. This algorithm is considered more efficient than Sondik's value iteration algorithm since it searches for witness points in smaller regions of the belief space.

The experimental results in [Cassandra, 1998a] indicate the witness algorithm is faster than other exact algorithms in practice over a wide range of small problems. However, there are also methods to improve the computational complexity of value iteration algorithms, which consequently yield a speed-up in the witness algorithm as well [Zhang and Zhang, 2001b; 2001a].

2.3.2.3. Incremental Pruning

The Incremental Pruning algorithm [Zhang and Liu, 1996; Cassandra *et al.*, 1997] achieves both simplicity and computational efficiency compared to the previous algorithms. A large number of the α-vectors are not useful in the parsimonious representation of the value function. This algorithm performs a dynamic programming backup and at each step it also prunes the dominated vectors.

The Bellman equation can be decomposed in the following steps:

$$Q^k_{a,z}(b) = \frac{R^a(b)}{|Z|} + \gamma Pr(z|a,b)V^{k-1}(b^a_z); \tag{2.16}$$

where $R^a(b) = \sum_{s \in S} b(s)R(s,a)$ is the immediate reward of taking action a from belief state b and b^a_z is the next belief state after taking action a from belief b computed by equation (2.5).

$$Q^k_a(b) = \sum_{z \in Z} Q^k_{a,z}(b); \tag{2.17}$$

$$V^k(b) = \max_a Q^k_a(b). \tag{2.18}$$

Cassandra [1998a] provides a detailed comparison of all exact algorithms. Zhang and Zhang [2001c] design a variation of Cheng's algorithm which performs a combination of point-based value updates and full value backup. This allows fewer number of expensive exact value backups while at the same time converging to the exact optimal solution.

The downside of the POMDP framework is that exact algorithms scale poorly. Exact POMDP solution methods are extremely demanding computationally even for fairly small state spaces. Even the best exact algorithms for POMDPs can be very inefficient in both space and time. Therefore, a huge research effort has been devoted to developing approximation techniques in this field.

2.4. Approximate POMDP Solution Methods

There are many approximation algorithms in the POMDP literature. These algorithms are mostly based on three general approaches for solving the planning problem:

- computing an approximate value function for exact belief space (see [Hauskrecht, 2000b] and [Pineau *et al.*, 2006] for reviews);
- computing the exact value function for compressed belief space (e.g. [Boyen and Koller, 1998; Poupart and Boutilier, 2000]);
- computing an approximate value function using compact belief space (e.g.[Rodriguez *et al.*, 1999; Roy *et al.*, 2005]).

Many approximation algorithms have been developed in each category. We mention just a few examples here. However, the focus of the thesis is on the first category.

2.4.1. MDP-Heuristics

There are a number of approximation techniques based on heuristics on the underlying MDP. For instance, the MLS heuristic [Nourbakhsh *et al.*, 1994], assumes that the agent is in the most likely state (MLS). This approach completely ignores the agent's confusion about which state it is in. The voting heuristic [Koenig and Simmons, 1998] weighs the vote for the best action in each state by the probability of being in that state. The popular QMDP heuristic [Littman *et al.*, 1995b] assumes that the POMDP becomes fully observable after taking one action. This heuristic first solves the underlying MDP and then, given any belief state, chooses the action that maximizes the dot product of the belief and Q values of state-action pairs:

$$QMDP(b) = \arg\max_{a \in A} \sum_{s \in S} b(s)Q(s,a) \tag{2.19}$$

These heuristics perform poorly if the belief state is close to uniform since their core assumptions are violated. This motivates choosing actions that decrease the entropy of the belief state in the hope that the heuristics above will perform better. The entropy can also be used to weigh two policies that trade off information gathering and exploitation [Cassandra *et al.*, 1996]. An alternative family of heuristics is based on simplified versions of the full dynamic programming update [Hauskrecht, 1997].

2.4.2. Grid-based Methods

Grid-based solution methods attempt to approximate the value function over the entire state space by estimating it on a finite number of belief states, contained on a chosen grid. Once a set of grid points has been chosen, an equivalent MDP can be constructed where the states are the grid points. This POMDP can be solved in polynomial time [Hauskrecht, 2000b]. Value functions over a continuous belief space can be approximated by values at a finite set of points along with some interpolation rule. Interpolation schemes should maintain the convex nature of the value function.

Grid-based approaches define a grid G containing a finite set of belief states. An interpolation-extrapolation function estimates the value at any point in belief space using the values at the grid points. The main problem with grid-based representations is that the number of required points increases rapidly with both the size of the state space and the resolution of the grid. In general, grid-based methods [Lovejoy, 1991a; Hansen and Zhou, 2003a; Brafman, 1997; Hauskrecht, 1997] differ along three main lines: (1) how the grid points are generated, (2) how the value function on the grid is estimated and, (3) how estimated values on grid points are generalized to the whole belief space.

Lovejoy [1991a] developed a fixed-grid method which selects grid points that are equally spaced in the belief simplex and calculates upper and lower bounds on the optimal value function. Using these bounds, we can get an approximately optimal policy and an error estimate. Using dynamic programming updates, the value function is estimated on the grid points. Lovejoy's choice of grid allows for an elegant and efficient interpolation method which estimates the value of arbitrary belief states based on the value of the grid points in the smallest sub-simplex containing this state. A very efficient interpolation technique based on triangulation assigns to non-grid belief states the convex combination of the values of nearby grid belief states. However, as the resolution increases, the number of grid points grows exponentially with the size of the state space.

Hauskrecht [1997] and Brafman [1997] proposed variable-resolution non-regular grids, which allow one to increase resolution in areas of poor accuracy by adding new grid points that are not necessarily equally spaced. This reduces the number of grid points while

achieving similar accuracy. However, because grid points are unevenly spaced, interpolation techniques are much more computationally intensive. The MLS heuristic can be thought of as a grid method with points at the corners of the belief space and a simple 1-nearest-neighbor interpolation (assuming an L1 distance measure) [Brafman, 1997]. Zhou and Hansen [2003a] proposed a variable-resolution regular grid that allows both fast interpolation and increased resolution only in the necessary areas. In general, for all grid-based methods the size of the grid can grow exponentially with the number of states of the underlying MDP. For this reason, the worst-case performance of grid-based methods is similar to exact methods. However, they work significantly better in practice.

2.4.3. Point-based Methods

Recently, algorithms have been proposed which take advantage of the fact that, for most POMDP problems, a large part of the belief space is not experienced by the agent and the actual belief states have a sparse probability distribution. Such approaches, which are known as point-based methods, consider only a finite set of belief points and plan for those points only. Plan generalization over the entire simplex is done based on the assumption that nearby points will have similar optimal values and actions. Point-based algorithms rely on the fact that performing many fast approximate updates often results in a more useful value function than performing a few exact updates. In this section, we present

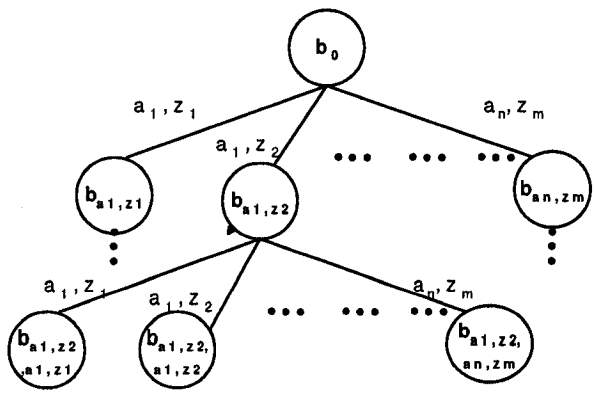


FIGURE 2.5. A forward search tree describing possible reachable belief points in two steps from the initial belief b_0 given n actions and m observations.

PBVI [Pineau *et al.*, 2003], HSVI [Smith and Simmons, 2004; 2005], and Perseus [Spaan

and Vlassis, 2005], three recent variations of the point based approach. These algorithms consider a starting belief state from which they try to predict the belief states that will be reachable by the agent. The point-based approach updates not only the values of the chosen belief states, but also their gradient. The value function is thus improved for all the belief state space and not only for the chosen belief states. These three methods differ in the approach they use for choosing belief states and the method they use to update the value function at the chosen belief states.

The point-based approach is an improvement of the variable grid-based approach. In the point-based approach, the belief states are sampled by starting in the initial belief state and simulating some (random) interactions of the agent with the POMDP environment. Hence, the sampled belief states have some chance of actually being reached by the agent during real interactions as well.

Point-based approaches can concentrate the computation on the attainable belief states. Thus, attainable belief states have more chance to be optimized. All the other belief states are also still assigned a value since point-based approaches keep an α-vector for each sampled belief state and not just a value.

One disadvantage of these methods is that they only optimize over a relatively small number of belief points, which is sometimes too small to give a good solution. Another drawback is that the α-vectors can become hard to manage in big state spaces because they still have as many elements as there are states in the environment.

2.4.3.1. Point-Based Value Iteration (PVBI)

Point Based Value Iteration (PBVI) is one of the point-based approaches and concentrates on planning for only a small set of belief states. It iteratively adds more points to the set in order to achieve a reasonable approximation of the value function. Empirically, PBVI [Pineau *et al.*, 2003], has been successful in finding high quality solutions very quickly. The PVBI algorithm [Pineau *et al.*, 2003] maintains a set B of reachable belief states from the starting belief state b_0. To do so, PBVI iteratively expands its set B by adding a new belief state for each belief state already in B, using a one-step stochastic exploration strategy. For

each belief state $b \in B$ and for each action $a \in A$, PBVI samples a state s from the distribution b, a resulting state s' from the distribution $T(s,a,s')$ and an observation z from the distribution $O(a,s',z)$. From these samples, it generates a new belief state b' by using the belief update rule. PBVI is an anytime algorithm since it interleaves phases of belief set expansion and value iteration.

The value iteration phase only tries to improve the values at the sampled belief states. Consequently, PBVI keeps at most one α-vector for each sampled belief state. After each value iteration phase, PBVI proposes a solution that improves as the number of belief states in B grows.

PBVI has had great success with solving problems that are an order of magnitude larger than POMDP problems solvable by exact methods. The full description of the PBVI algorithm is presented in Algorithm 2.

Algorithm 2 Point Based Value Iteration (PBVI)

Input: B_{init}, Γ_0, N, T
$B = B_{init}$
$\Gamma = \Gamma_0$
for N expansions **do**
  for T iterations **do**
    Γ = Point-Based Value Backup(B,Γ)
  end for
  B_{new} = Expand (B,Γ)
  $B = B \cup B_{new}$
end for
Return Γ

The backup operator creates $|A||Z||\Gamma_{t-1}|$ projections, but the expensive pruning operator needed in the exact algorithms is not necessary for PBVI. The algorithm in one expansion takes polynomial time in the number of states,actions,observations,α-vectors, and the number of belief points at that expansion ($O(|S|^2|A||Z||\Gamma_{t-1}||B|)$).

The point-based value backup presented in Algorithm 3, is a fundamental operation for all point-based methods.

Algorithm 3 Point-Based Value Backup

INPUT: Value-function Γ_{t-1} *and Belief point set B*
OUTPUT: Value-function Γ_t
for all $a \in A$ **do**
  for all $z \in Z$ **do**
    for all $\alpha' \in \Gamma_{t-1}$ **do**
      $\alpha^{a,z}(s) = \gamma \sum_{s' \in S} T(s,a,s') O(a,s',z) \alpha'(s')$
      $\Gamma_t^{a,z} \leftarrow \alpha^{a,z}$
    end for
  end for
end for

for all $b \in B$, $s \in S$ **do**
  $\alpha_b \leftarrow$ arg $\max_{a \in A} [\sum_{s \in S} R(s,a) b(s) + \sum_{z \in Z} \max_{\alpha \in \Gamma^{a,z}} [\sum_{s \in S} \alpha(s) b(s)]]$
end for
if $\alpha_b \notin \Gamma_t$ **then**
  $\Gamma_t \leftarrow \alpha_b$
end if
Return Γ_t

2.4.3.2. PERSEUS

Spaan and Vlassis 2005 described a randomized point-based value iteration algorithm called *PERSEUS*. Unlike PBVI, this algorithm guarantees monotonic value function improvement from one iteration to the next as the algorithm progresses. The value function is derived from the optimal value of a fixed and relatively large set of belief points.

Perseus operates on a set of belief states which are gathered by simulating random interactions of the agent with the POMDP environment. At the same time, it uses the point-based backup operator for a small number of times instead of applying it repeatedly on sets of points of different sizes. The key idea is that in each value iteration step it is possible to improve the value of all points in the belief set by only updating the value and its gradient on a subset of the points. This allows the algorithm to compute efficiently value functions that consist of only a small number of vectors relative to the belief set size. Their approach is considerably faster than PBVI and the authors have efficiently solved a large robotics application.

2.4.3.3. Heuristic Search Value Iteration (HSVI)

Another algorithm based on the point-based value iteration approach is the *HSVI* algorithm [Smith and Simmons, 2004; 2005]. HSVI maintains both upper and lower bounds on the optimal value function. The lower bound is a set of α-vectors like for PBVI and the upper bound is represented as another convex hull defined with a set of belief points B.

Initially, the lower bound is initialized with only one α-vector, representing the worst possible case if the same action is applied indefinitely. The upper bound is initialized with the solution of the underlying MDP. The bounds are updated at specified belief points iteratively. For the lower bound, the update at a belief state b consists of adding a new α-vector defining the value function at b, as in the PBVI algorithm. For the upper bound, the update at a belief state b consists of adding a new belief point in the set defining the convex hull. The belief points are selected using the search tree of the reachable beliefs from the initial belief state b_0. The search is directed by using the lower and upper bounds.

The first version of HSVI [Smith and Simmons, 2004] evaluates $V(b)$ by computing the exact projection of b onto the convex hull of the points in B, which involves solving a linear program. Each upper bound update requires several such projections, which is time consuming. In the second version of HSVI [Smith and Simmons, 2005] an approximate projection into the convex hull suggested by [Hauskrecht, 2000b] is used instead. To approximately project into the overall convex hull, the algorithm runs this operation for each interior point of B and takes the minimum value, requiring overall $O(|B||S|)$ time. Empirically, Smith and Simmons show that the approximate projection speeds up the upper bound updates by about two orders of magnitude.

2.4.4. History-Based Methods

When the model of the environment is unknown (i.e., the agent does not know the transition function, the observation function, and the reward function), it is possible to construct POMDP policies based on the history of past actions and observations instead of the belief state [Chrisman, 1992; Mitchell, 2003; McCallum, 1995b; Dutech, 2000; Shani, 2004]. These algorithms are able to construct good policies based on the agent's

past, but they usually need a lot of data. MEDUSA algorithm [Jaulmes *et al.*, 2005] can be considered as a decision making algorithm in model-free POMDPs. This algorithm incrementally learns a POMDP model using oracle queries. MEDUSA is more flexible with the prior knowledge and data requirement compare to other history-based methods due to selecting when to do queries.

McCallum's utile suffix tree algorithm [McCallum, 1995b] uses the agent's past experiences to construct a simplified tree representation of the state space. This tree groups together past experiences that have similar values. Afterwards, based on its current past actions and observations, the agent can go down the tree and find the corresponding abstract state (a leaf of the tree), which contains the Q-values defining the agent's policy. McCallum was able to successfully apply this technique to a large application of simulated driving in a road task.

2.4.5. Policy Search Methods

There are other methods for solving POMDPs which directly search the space of policies rather than using the value iteration method [Ng and Jordan, 2000; Hansen, 1998; Sutton *et al.*, 1999a; Aberdeen, 2005; Bernstein *et al.*, 2005]. The most popular method in this class is known as *policy iteration*. Policy iteration is an iterative two-phase algorithm. In the first part, *policy evaluation*, it tries to evaluate a policy being generated over the entire belief space. In the second part, *policy improvement*, it attempts to improve this policy to a better approximation of the optimal policy. Most of the policy search techniques use a finite state machine to represent a policy. Many algorithms have been developed to search in the space of policies [Hansen, 1998; Hansen and Zhou, 2003b; Poupart and Boutilier, 2003a; Aberdeen and Baxter, 2002; Meuleau *et al.*, 1999; Braziunas and Boutilier, 2004]).We do not describe them in detail here, as they go beyond the scope of this thesis.

2.5. Summary and Conclusion

In this chapter we discussed Partially Observable Markov Decision Processes (POMDPs), an expressive framework for sequential decision making under uncertainty. POMDP policies handle uncertainty well. However, solving POMDPs is very demanding computationally. We reviewed the basic concepts for POMDP exact solution techniques, presented some of these methods, and described the sources of intractability for exactly solving POMDPs. Many approximation techniques have been proposed by different researchers to overcome the scalability of POMDP solution techniques.

Point-based approximation methods aim to reduce the curse of history defined in this chapter. They are promising with respect to both scalability and solution quality. However, there is still room for improvement. One of the issues in designing point-based methods is the point selection mechanism they adopt. Ideally, we seek techniques which provide a guaranteed good quality solution with small number of points. We will return to this issue in Chapter 4. There are a number of other interesting directions to point-based approximation. Combining point-based methods with approaches that tackle the curse of dimensionality in particular is very important in overcoming the computational complexity of decision making under uncertainty.

CHAPTER 3

Predictive Representations of Dynamical Systems

Chapter Outline

In Chapter 2 we discussed POMDPs as a general framework for representing dynamical systems. Unfortunately, POMDPs are difficult to solve. It is also difficult to learn the model from data. Predictive State Representations (PSRs) [Littman *et al.*, 2001; Singh *et al.*, 2004] have been proposed recently as an alternative representation for environments with partial observability. The representation is rooted in actions and observations, so it holds the promise of being easier to learn than POMDPs. In this chapter, we review PSRs. In Section 1 we introduce the PSR model and summarize the development of the PSR framework through a mathematical construct called *system dynamics matrix*. We also explore the characteristics of the model and explain the concepts required for the analysis of this model. Section 2 discusses different types of predictive models, representing extensions the original definition of PSRs. In Section 3 we review models related to PSRs. Section 4 summarizes the chapter.

In real world systems, knowledge of the future is based on predicting how the world will be in terms of some known features. A predictive model is used to predict how a

change (e.g., taking an action) will affect other conditions (e.g., prediction of observations) as the system evolves in time. Therefore, the model needs to maintain and update its predictions at each point in time. Predictive State Representations (PSRs) have been developed to provide a learnable and maintainable representation of the knowledge of the system. Predictive models hold the promise of providing a self verifiable representation due to the mathematical structure of the model states. This is one of the most distinctive characteristics of predictive representations.

The PSR representation, which was inspired by the earlier work on Observable Operator Models (OOMs) [Jaeger, 1998] and diversity-based representation [Rivest and Schapire, 1994], uses predictions of events in the future to describe a dynamical system.

An important issue for automated agents concerns learning the model. Learning models of dynamical systems under uncertainty has been widely studied for different frameworks. In this thesis, we do not consider this problem and throughout this work we assume that the world model together with its parameters are given.

3.1. PSR Model Specification

PSRs are based on testable experiences. The notion of *test*, used in the definition of PSRs, carries the central idea of relating states of the model to verifiable and observable quantities. A test is an ordered sequence of action-observation pairs $q = a_1 z_1 ... a_k z_k$. The length of a test is the number of action-observation pairs it contains. The length can be zero, in which case it is called a *null test*. We denote a null test by ε.

The *prediction* for a test q, specified as above, is the probability of the sequence of observations $z_1, ..., z_k$ being generated, given that the sequence of actions $a_1, ..., a_k$ was taken. If this observation sequence is generated, we say that the test succeeds. It must be noted that the probability of a test succeeding depends on the history of past experiences. A history can be viewed as a test with the only difference being that it starts at the beginning of time ($t = 0$) while a test can start at any arbitrary time in the life of the agent. A history can also be of length zero, which specifies the beginning of time or the initial condition in a system. Such a history is called a *null history* and denoted by ϕ. The prediction for a

test q given the prior history $h = a_1^h z_1^h ... a_t^h z_t^h$ is denoted by $p(q|h)$. This is the conditional probability of a test q being successful given that the test is performed after history h:

$$P(q|h) = P(z_{t+1} = z_1, ..., z_{t+k} = z_k | h, a_{t+1} = a_1, ..., a_{t+k} = a_k) \tag{3.1}$$

Therefore:

$$P(q|h) = \frac{P(hq)}{P(h)} \tag{3.2}$$

The null test is always successful, or has probability of 1, from any history that can possibly be experienced in the system. The prediction for a test q can be updated as the agent accumulates more experience with the world. This update can be done incrementally after each action-observation as:

$$P(q|haz) = \frac{P(azq|h)}{P(az|h)} \tag{3.3}$$

For any set of tests Q, its prediction is specified by a vector of probabilities, one for each test member $q_i \in Q$:

$$P(Q|h) = [P(q_1|h), ...P(q_{|Q|}|h)] \tag{3.4}$$

A set of tests Q is a PSR of a dynamical system if its prediction, which is called the *prediction vector*, $P(Q|h)$, forms a sufficient statistic for the system after any history h, i.e., if a prediction for any test q at any history h can be computed based on $P(Q|h)$.

$$P(q|h) = f_q(P(Q|h)) \tag{3.5}$$

where $f_q : [0,1]^{|Q|} \rightarrow [0,1]$. It is important to note that f_q, which is called *projection function*, does not depend on the history h. The update equation can be written as:

$$P(q|haz) = \frac{f_{azq}(P(Q|h))}{f_{az}(P(Q|h))} \tag{3.6}$$

This equation suggests that all the model needs to do is to update the prediction vector $P(Q|h)$, at any given history h. The elements of Q are called *core tests*. The PSR model requires keeping the prediction vector updated at each point in time; this represents the state of the system at each time. If we have the projection function f_{az} for all one-step tests,

and projection function f_{azq_i} for all one-step extensions to the core tests, we can update the state of the PSR model.

The model parameters are described by the projection function of the set of *extension tests*: $\{azq_i, az|\forall q_i \in Q, a \in A, z \in Z\}$. The size of the model, or the number of extension tests, is proportional to the size of the set Q. The number of core tests, $|Q|$, is called the *dimension* of the model. The PSR representation of a dynamical system has at most a number of core test equal to the number of hidden states in the POMDP representation [Littman *et al.*, 2001]. In fact, the PSR model is potentially more compact than the corresponding POMDP.

The type of projection function f_q defines the type of a predictive state representation. If f_q is linear, the PSR model is called linear as well, otherwise it is called non-linear. Rudary and Singh [2004] introduced a case of non-linear PSRs, but this category has received little attention to date. We briefly discuss variations of PSRs in the following sections.

3.1.1. Linear-PSR

A *linear-PSR* is a PSR in which there exists a projection vector m_q for any test q such that

$$P(q|h) = P(Q|h)^T m_q \tag{3.7}$$

In this case, Equation (3.6) for updating the prediction vectors becomes:

$$P(Q|haz) = \frac{[P(azq_1|h)...P(azq_k|h)]}{P(az|h)} \tag{3.8}$$

A linear PSR model consists of:

- A : finite set of actions;
- Z : finite set of observations;
- Q : finite set of selected tests $\{q_1, q_2, ..., q_k\}$ (core tests);
- m^{az} : weight vectors for projections of one-step tests, defined for each action $a \in A$ and each observation $z \in Z$;

- m^{azq_i} : weight vectors for projections of one-step extensions of core tests, defined for each action $a \in A$, each observation $z \in Z$ and each core test $q_i \in Q$.

The PSR model of a POMDP can be related to the POMDP model itself through the definition of *outcome functions*. Littman et al. [2001] define an outcome function $u : Q \rightarrow [0,1]^n$ mapping tests into n-dimensional probability vectors defined recursively as:

$$u(aoq) = (T^a O^{a,z} u(q)) \tag{3.9}$$

$$u(\varepsilon) = e_n$$

where T^a is the transition matrix for action a in the POMDP model, $O^{a,z}$ is a diagonal matrix of size $n = |S|$(the number of unobservable states in the POMDP) and each element $O^{a,z}_{ii}$ denote the probability of observation z after taking action a and arriving in state s_i upon taking action a , ε represents the null test, and e_n is the $(n \times 1)$ vector of all 1s. Each component $u_i(q)$ indicates the probability of the test q when its sequence of actions is applied from state s_i.

Definition: ([Littman *et al.*, 2001]) A set of tests $\{q_1, ..., q_n\}$ are called *linearly independent* if and only if their outcome functions $u(q_1), ..., u(q_n)$ are linearly independent vectors.

LEMMA 1. *([Littman* et al., *2001]) The outcome vectors of the tests in Q can be linearly combined to produce the outcome vector for any test.*

THEOREM 1. *([Littman* et al., *2001]) For any environment that can be represented by a finite POMDP model there exists a linear PSR with number of tests no larger than the number of states in the POMDP model.*

PSRs can potentially represent dynamical systems more compactly than POMDPs, but they still provide sufficient information to determine the behavior of any system that can be modeled in the POMDP framework. Figure 3.1 depicts the matrix containing the outcomes of all tests. This matrix is called the U-matrix in the PSR literature. The tests in this matrix

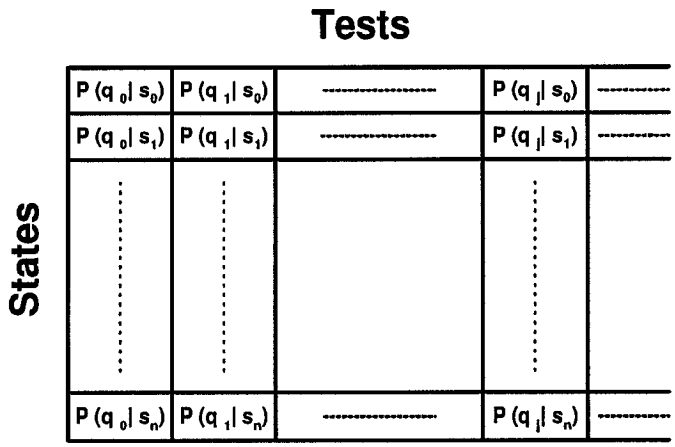


FIGURE 3.1. Dynamics of a system represented in the outcome matrix U.

are sorted by their length and then lexicographically within the same length. A complete model of a system should be able to predict the probability of an arbitrary experience. The POMDP model can make such predictions using its internal state definition as follows:

$$P(q|h) = b_h^T (T^{a_1} O^{a_1,z_1} ... T^{a_l} O^{a_l,z_l}) e_n \tag{3.10}$$

where b_h^T is the transpose of the belief state vector after history h. This equation can be rewritten as:

$$P(q|h) = b_h^T m_q. \tag{3.11}$$

Using the definition of outcome function u, the set Q of core tests can be found by searching for the maximum number of linearly independent tests. This algorithm, which is presented in Algorithm 4 [Littman *et al.*, 2001], incrementally finds all core tests in an iterative fashion, given the POMDP model of the environment.

3.1.2. System Dynamics Matrix

The *system dynamics matrix* (SDM) is a mathematical construct that captures the behavior of a discrete-time dynamical system completely. Each entry of the matrix is the conditional prediction probability of a test given a history. This matrix, which we denote by D, is the underlying structure for learning PSRs and discovering the core tests from data. The SDM was first introduced by Singh et al. [2004] to explain the predictive state

Algorithm 4 Generating core tests from a POMDP model

$i \leftarrow 0$
$Rank(i) \leftarrow 0$
$Q \leftarrow \emptyset$
repeat
  $i \leftarrow i+1$
  $U_i \leftarrow$ {all one-step extensions of Q} $\cup Q$
  $Rank(i) \leftarrow$ rank of U_i
  $Q \leftarrow$ { tests corresponding to linearly independent columns in U_i}
until $Rank(i) = Rank(i-1)$
Return Q

representation purely based on observable data. The outcome matrix U defined in the previous section still relies on the unobservable underlying state space S. Although learning the PSR model from the U-matrix is exact, the entries of this matrix are computed given the POMDP model parameters, action transition functions and observation functions. Such a model may not be available, or the system may not be a POMDP.

In order to learn PSRs from observable data without a POMDP model, the rows of the SDM should correspond to the possible histories of actions and observations that the system can generate: $H = \{\phi, h_1, ...\}$ and its columns must indicate the agent's possible experiences in the future: $Q = \{q_0, q_1, ...\}$. Each entry D_{ij} of the matrix is defined to be $P(q_j|h_i)$. Although the SDM is an infinite dimensional matrix, we can express it succinctly using the following theorem:

THEOREM 2. *([Singh* et al., *2004]) Any dynamical system that can be modelled as a POMDP with n states has a system dynamics matrix of rank at most n.*

The system dynamics matrix forms the basis for PSR learning algorithms presented in [James and Singh, 2004a; McCracken and Bowling, 2006; Wiewiora, 2005; Wolfe *et al.*, 2005; James and Singh, 2004b]. The first row of this matrix which contains the prediction for all tests at the beginning of time, given only the null history ϕ, is known as *system dynamics vector*. This vector defines a full specification of the dynamical system by capturing the prediction of all tests at the beginning of time, t_0. These predictions however are not independent of each other, and there are some constraints on them forced by the laws of

probability. The linearly independent columns and rows of D correspond to core tests and

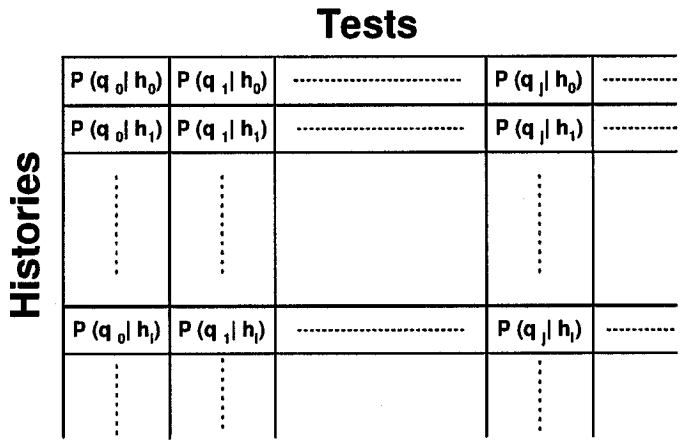


FIGURE 3.2. System Dynamics Matrix (SDM)

core histories respectively. It is obvious that these sets are not unique; however there is no evaluation or analysis available on choosing among sets of tests or histories except for a trivial preference on the length of the tests (i.e selecting shorter tests and histories which are linearly independent is preferred to those that are longer) for ease of computation. For further reading and detailed discussion on the SDM, see [James, 2005].

In automata theory, a similar structure to SDM, called *Hankel matrix*, is used to learn a special case of automata called *multiplicity automata* [Shlitzenberger, 1961]. We briefly review this model later in this chapter due to its similar characteristics to PSRs and its similar use in decision making.

3.1.3. PSR Model Learning

The input to the problem of learning dynamical systems is a sequence of experiences with the system. The dynamical system determines what the learner is able to observe at any given time, conditioned on the action taken at that time. The important question for learning a particular dynamical system is how to go from observations to a model that will support predictions.

There are two major parts to PSR model learning: finding the set of core tests Q (known as the discovery problem), and learning the weight vectors, or projection vectors m_{ao} and m_{aoq_i} (known as the parameter learning problem).

The first PSR learning algorithm proposed by Singh et al. [2003] tried to solve the parameter learning problem using an approximate gradient descent algorithm. Their algorithm works relatively well on some domains. This approach assumes that the set of core tests is provided beforehand. However, knowing the core tests is a big assumption and not very reasonable.

Solving the discovery problem together with learning parameters has been attempted by James and Singh [2004a] for system with a reset action. In this kind of system we assume the existence of a reset action, which the agent can use to get back to the initial starting condition, reproduce histories and generate multiple samples. The system dynamics matrix is estimated by computing the maximum likelihood of each of its entries. The core tests are found by searching for the linearly independent columns of the SDM similarly to the algorithm for core test discovery (except that the matrix D is used instead of U).

Wolfe et al. [2005] presented a modified algorithm for learning and discovery for PSRs in systems without reset, called the *suffix-history* method. In this approach, the histories with identical suffixes are grouped together for counting. All PSR discovery methods suffer from the fact that generating core tests is very much related to computing the SDM. Estimating the prediction probabilities for each entry of the SDM usually requires a large number of test/history samples. Moreover, computing SDM entries approximately makes the computation of the rank of this matrix numerically unstable. Recently, McCracken and Bowling [2006] designed an online learning algorithm for PSRs called *constrained gradient* which uses a gradient descent approach to estimate predictions of tests in an online fashion. This approach makes more efficient use of data (in the form of the agent experience with the world) to discover core tests and also avoids the rank estimation in the original PSRs' discovery algorithm [James and Singh, 2004b].

3.2. Extensions of the linear PSR model

Since the discovery of the predictive state representation framework, there have been a number of other predictive representation models based on similar ideas. We review these models in this section.

3.2.1. Non-linear PSRs

In non-linear PSRs, the prediction for a test is not a linear function of the predictions for the core tests, but an arbitrary function of it. In this case, there may exist a smaller number of core tests such that the prediction for any test q and any history h can be determined by an arbitrary function on them. Equations (3.2) and (3.6) still hold for this type of PSRs. Figure 3.3 illustrates the *float-reset* example which was introduced in [Littman *et al.*, 2001]

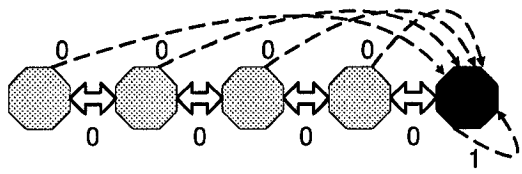


FIGURE 3.3. Float-Reset problem

and frequently used in the PSR literature after that. In this example there are five nominal states, two actions (float and reset) and two observations (0 and 1). The dynamics of the system is such that a float action moves the agent to one of its two neighbors uniformly randomly and the reset action takes the agent to the rightmost state. The agent observes 1 only when it takes the reset action from the rightmost state. The non-linear PSR model only needs two core tests to represent the system as opposed to the PSR model which requires five core tests.

We mentioned that non-linear PSRs can be potentially more compact than the corresponding linear PSRs. However, to date there is very little research on this model. In general, we know that nonlinearity is at the heart of many of the interesting dimensionality reduction methods. However, along with this attractive feature, the complexity of the models increases, and so they are also more difficult to analyze.

3.2.2. Event-PSR (EPSR)

Event-PSR (EPSR) is a special kind of linear PSR in which tests are of the form of a sequence of primitive actions which leads to an observation, rather than a sequence of primitive action-observation pairs. This type of PSR is very close to the notion of *options* defined in [Sutton *et al.*, 1999c; Precup *et al.*, 1998; Precup and Sutton, 1998] for MDPs. We say that an EPSR test $q = a_1 a_2 ... a_n z_n$ is *successful* when upon taking the sequence of actions $a_1 a_2 ... a_n$ at history h, the agent makes the observation z_n at time n. The EPSR model includes all one-step tests (which are included in previously discussed PSR models as well).

Finding the core tests of an EPSR model is possible by using the system dynamic matrix. The prediction of tests becomes: $u(q) = T^{a_1} T^{a_2} ... T^{a_n} O^{a_n z_n}$. Therefore, the observations along the way in an ordered sequence of actions do not matter and the size of the matrix grows less quickly ($O(|A|^k|O|)$ instead of $O(|A||O|)^k$ for k-step tests).

Rudary and Singh [2004] showed an example of a deterministic system for which EPSR models lead to exponential compression over POMDP models. Rafols et al. [2005] developed several interesting examples that exhibit a great amount of regularity and showed how EPSR models can capture this type of structure.

3.2.3. Temporal-Difference Networks (TD-networks)

TD-networks [Sutton and Tanner, 2004] are a generalization of PSRs which represent networks of interrelated predictions in which each prediction can be used for computing other predictions. TD-networks are a combination of a version of temporal difference prediction [Sutton, 1988] with predictive state representation. The networks consist of a set of nodes, each representing a single scalar prediction. The nodes are linked together based on the temporal difference relationship among the predictions. These links, or so called *question-network or what-network*, represent what the network can predict about the data generated from the environment as a sequence of action-observation pairs. The nodes in the network can be thought of as predictions of different semantics. For example, a node could represent the conditional prediction of an observation given an action (like PSRs

predictions), while another could represent the prediction of the expected value of another node, or the prediction of an observation k time steps ahead.

The state of the system is represented by a vector of predictions of all nodes in the network. For instance, if the network consists of n nodes $y^1 ... y^n$, then at each point in time it should maintain a vector, $y_t = (y_t^1, ..., y_t^n)$ and update this vector according to a function called the *answer-network*. TD-networks combine two developed abstraction technologies: predictive state representations for abstracting over states, and the options framework for abstracting over time. The idea of TD-networks is still new and the strengths and limitations of this model have yet to be fully determined (see [Tanner and Sutton, 2005] for recent developments). Sutton and Tanner [2004] suggest that TD networks can be used for learning PSRs. Their preliminary experiments show promising results, but this idea is still under investigation.

3.2.4. Memory-PSRs (m-PSRs)

The idea behind the *m-PSRs* is to use both the predictions about the future events and a memory of the past to represent states. It can be thought of as a combination of PSRs and history-based representations. The foundation of the model is based on a variation of the system dynamic matrix defined in [Singh *et al.*, 2004].

The development of this model is done by using a special partitioning of the system dynamics matrix (D) into equivalence classes. This partitioning is based on the notion of *memory* as a fixed length history of past action-observation pairs. In this construct, two histories are considered equivalent if they have identical action-observation pairs in their last k steps, for a predetermined amount of time k. Therefore, the rows of D are partitioned into m parts $D_1, ..., D_m$ such that they describe the partition of the histories into m disjoint groups. Each history of the dynamical system is related to one of the memories $\mu_1 ... \mu_m$ by identifying its last k action-observation steps. Each partition specifies a sub-system S_i. Therefore, each partition D_i has a subset of core tests Q_i corresponding to the linear dimensions of the sub-matrix S_i and its own model parameters. It is important to note that the histories in the partitions are disjoint, but the core tests are not necessarily disjoint.

The state of an m-PSR model at a history h is defined by a pair $(\mu(h), P(Q^{\mu(h)}|h))$. The memory μ is related to history h and the prediction for core tests $Q^{\mu(h)}$ at this history. The number of parameters of a memory-PSR model can be much larger than that of the corresponding PSR model. However, in practice only a small number of core tests in each partition may be required, which makes the total number of parameters smaller than for linear PSR in some cases as illustrated by James et al. [2005]. It must be noted that selecting the appropriate memory length is essential for this construct. The experimental results in [James and Singh, 2005; James *et al.*, 2005] show that an accurate m-PSR model can be built successfully from samples, whereas PSR model learning is not very accurate. However, using the m-PSR model is not advantageous when the number of model parameters significantly increases.

3.2.5. Predictive Linear-Gaussian Model

In many real world domains, one (or more) of the state, action, or observation spaces are not discrete. Handling continuous spaces is more difficult than the discrete case for obvious reasons. The predictive linear Gaussian (PLG) model has been recently proposed by Rudary et al. [2005] to model dynamical systems with continuous observations. The model was developed for uncontrolled dynamical systems (i.e., no actions) with some restrictive assumptions about the distribution of observations: the joint distribution of observations is multivariate Gaussian, and the distribution of each future observation can be completely determined by the next n observations (for a finite number, n).

This model maintains the parameters μ_t and Σ_t representing the distribution of the observation z_t at time t as its state, and updates these state parameters as a new observation z_{t+1} is made at history h_t. Based on the properties of the predictive model and its constructive assumptions about the observations (the distribution of the next n observations captures all information about the distribution of all future observations), μ_t and Σ_t are sufficient statistics for history h_t, which makes it possible to resolve the problem of having infinite memory without introducing new hidden variables (as in Kalman filters).

The Predictive Linear Gaussian model improves the traditional linear dynamical system (LDS) models by using a predictive representation of state. This improves the parameter estimation and uses fewer parameters [Rudary and Singh, 2006; Wingate and Singh, 2006]. However,it is not always an alternative approach to LDS, because of its restrictive assumptions.

3.3. Existing Related Frameworks

Historically, there have been two dominant approaches to modeling dynamical systems: state-based models such as POMDPs and history-based models such as k-order Markov models. We have considered the POMDP framework in detail in Chapter 2, and in this chapter we discussed the relationship between POMDPs and predictive state representations. Here we briefly describe other models, including history-based representations and other predictive models, some of which can be viewed as inspiration and foundation for developing PSRs. We discuss their relation to PSRs.

3.3.1. History-based Models

History-based representations [Platzman, 1997; McCallum, 1995b] are not as general and as powerful as PSRs and POMDPs since in many systems, there are long sequences of experiences which provide useful information about the system. For instance, imagine we need a model for localization of an agent walking around a town. Assuming that the agent cannot identify generic buildings and shops it walks into and distinguish them from the similar places in different regions of the town, it needs to remember useful information, such as the signs at a particular exit of the roundabout, which provide the knowledge about that particular area. Therefore, the agent needs to remember where in the history of its interactions it saw the signs. In other words, it has to memorize all the steps taken from its origin to be able to localize itself at each point in time. Obviously, this could lead to maintaining an infinite length history of its experiences in order for the agent to distinguish generic places, which is infeasible. If there was a way to only remember useful information in the past and forget irrelevant parts, the agent would have no problem identifying its

location. Unfortunately, the agent cannot be selective in what to remember, since there is no clear way to measure the usefulness of past information as the agent behaves.

A general history-based method may need to remember an arbitrary amount of experience in order not to forget important information. The PSR approach to this type of problems contains a set of experimental tests that suffices to localize the agent. A model with too many degrees of freedom becomes unmanageable, while a too constrained one becomes unreliable. It is necessary to mention that the agent does not need to perform all possible scenarios in future to gain useful information. Rather than explore possible futures of any real diversity, it needs to predict only a fairly narrow set of key experiences.

In the history-based model, the agent remembers as much of its history as it can. Identifying the real, hidden, states of the world using any finite length window of the history, however, is not possible in some cases. While disambiguating the state may require the agent to have infinite memory of the past, some approaches or special case problems have been designed to avoid this problem. For instance, using variable length finite history windows, McCallum's instance-based state identification [McCallum, 1995a] resolves perceptual aliasing with variable-length short term memory.

3.3.2. Observable Operator Models

One of the predictive state models of stochastic systems is the *observable operator model (OOM)* [Jaeger, 2000]. The original OOM model was designed for uncontrolled dynamical systems. It was shown to be more powerful, and able to better capture the properties of certain stochastic processes than *hidden Markov models (HMMs)* (an HMM models the random walk of an uncontrolled dynamic system, and as such can be seen as a POMDP with only one action). An OOM is a triple (R^m, τ_a, w_0), where w_0 is a state vector in R^m and τ_a $(a \in Z)$ are linear operators satisfying the following conditions:

(i) sum of all components of w_0 is 1;

(ii) τ_a has columns whose entries sum up to 1;

(iii) for all tests $t_0 \ldots t_k$, the sum of all components of $\tau_{a_k}, \ldots, \tau_{a_0} w_0$ are nonnegative.

Jaeger developed some other variations of OOMs including interpretable OOMs and input-output OOMs [Jaeger, 1998], for controlled dynamical systems which have many similarities with PSRs. James [2005] showed that, in the case of uncontrolled dynamical systems, linear PSRs and interpretable OOMs are equivalent, while in the case of controlled systems, linear PSRs are more general.

3.3.3. Finite State Automata and Multiplicity Automata

A Finite State Automaton model (FSA) [Hopcroft and Ullman, 1979], is a five-tuple (S, Σ, δ, s_o) where S is the set of states of the FSA, Σ is the set of input symbols, δ is the transition function between states and s_0 is the initial state of the FSA. Finite state automata are learned by observing the result of sequences of actions. Let K be a field and f be a mapping from Σ^* to K. A matrix F called a *Hankel matrix* can be associated to the function f such that the rows and columns of this matrix correspond to strings $x, y \in \Sigma$, and the entry $F_x(y)$ is the value of $f(xy)$ where xy denotes the concatenation of string x and y. In general, F is an infinite matrix. Similarly to the system dynamic matrix, if strings are restricted to length at most n, the size of F is exponential in n, independently of the complexity of the function it represents.

Multiplicity automata are a generalization of deterministic and stochastic automata. As such, they can still take advantage of some of the learning methods from automata theory [Beimel *et al.*, 1996]. Learning is based on a search through the Hankel matrix for finding a maximal number of linearly independent rows and columns. Multiplicity automata have been used as an alternative representation for POMDPs. The algorithm of Even-Dar et al. [2005], for using multiplicity automata in POMDP planning uses the core-belief construct which we will introduce in the next chapter. Of course, this algorithm is advantageous compared to POMDP planning whenever there is a structure which leads to a small size model, just as the case with PSR planning.

3.3.4. Diversity-Based Representation

Rivest and Schapire [1994] considered the problem of inferring the structure of a deterministic finite state environment by experimentation. The learner is assumed to have no

a priori knowledge of the environment other than knowing how to perform a set of actions and knowing what observations are possible. Both the transitions and observations are deterministic. The determinism is encoded by specifying that all entries of the transition matrices $T^a(\forall a \in A)$ must be either 0 or 1. Similarly, all entries of the observation matrices $O^{az}(\forall a \in A, z \in Z)$ must be 0 or 1. Their algorithm effectively infers deterministic finite automata from input-output behavior in the absence of a means of resetting the machine to a start state.

Predictions in the diversity representation are for tests called *e-tests*, defined the same way as in EPSR [Rudary and Singh, 2004]. However, since the systems in question are deterministic, predictions for e-tests are always 0 or 1. Rivest and Schapire [1994] defined two e-tests q_1 and q_2 to be *equivalent* if, for every nominal-state, the prediction for q_1 is equal to the prediction for q_2. The number of equivalence classes is called the diversity of the system and the state vector of the diversity representation contains one prediction for each equivalence class. In general, diversity can be larger or smaller than the number of states. For a model with diversity d, the state at history h is written as: $[p(q_1|h), ..., p(q_d|h)]$, where q_i is the ith equivalence class and $p(e_i|h) = p(q|h)$ for any e-test q in equivalence class e_i. Rivest and Schapire also showed strict bounds that relate the diversity of the system to the number of nominal states in a minimal deterministic automata on that system.

THEOREM 3. *([Rivest and Schapire, 1994]) For a dynamical system that has a minimal deterministic model with n nominal-states, the diversity d of the system is constrained by:* $log_2(n) \leq d \leq 2^n$

The diversity-based representation uses permutation matrices when mapping an old state vector to a new one after executing an action, whereas a PSR uses the more general SDM matrix to apply the update function. The diversity-based representation is one of the inspirations for the development of PSRs and especially EPSRs. However, this representation is limited since it has the strong assumption of determinism.

3.4. Summary and Conclusion

In this chapter, we presented the basic theory and algorithms for using predictive representations of state. PSRs maintain the key characteristic of using only observable, verifiable quantities in the state representation. The state of the model must satisfy the Markov property: when given the state, all possible future outcomes are conditionally independent of the history. Therefore, the prediction of every test is a function of the prediction vector alone. The theoretical analysis of the predictive models, their possible extensions, and their possible augmentation in other interesting frameworks are still under investigation.

Another aspect of PSRs is learning the structure of the model. The learning algorithms developed for PSRs so far seem to be computationally as difficult as POMDP model learning methods in general [James, 2005]. This thesis does not address the PSR learning problem. James [2005] has a complete description of the PSR learning problem and the approaches to solve this problem.

We will come back to PSRs in Chapters 5 and 6. The question of most interest to us in this thesis is how to plan with a PSR model. Exact or approximate planning methods for POMDPs can be extended to work for PSRs with some modifications, as we describe in Chapter 6. Predictive representations were designed with the hope to represent dynamical systems in a more compact way than POMDPs. This is the motivation for our work on PSRs model minimization [Izadi and Precup, 2005a] presented in Chapter 5, which looks at the characteristics of domains under which such reduction is possible.

CHAPTER 4

Belief Selection Strategies in Point-Based Approximation Methods for POMDPs

Chapter Outline

In this chapter we focus on the powerful class of point-based algorithms. We address the issue of dynamically generating a good ordering of beliefs in an efficient way, and explore the point-based value iteration algorithm in combination with a number of belief point selection heuristics. Section 1 proposes a new method for finding the optimal value for the basis of the reachable belief space. Section 2 presents a selection strategy based on reachability analysis from a given initial belief state. Section 3 points out the problem of exploration versus exploitation which impacts the efficiency of PBVI. It also discusses the impact of the structure of the belief space on point selection schemes. Section 4 describes an approach that takes into account the current estimate of the value function for prioritizing beliefs in the belief set expansion phase. Section 5 contains empirical results illustrating how the performance of point-based value iteration varies depending on these selection criteria. Concluding remarks are presented in Section 6.

Recent research on POMDP approximation has been devoted to the algorithms that take advantage of the fact that for most POMDP problems, a large part of the belief space is never experienced by the agent. There has been some work recently on more efficient

techniques for approximately solving POMDPs, as discussed in Chapter 2. In particular, point-based planning algorithms for solving partially observable Markov decision processes have demonstrated that a good approximation of the value function can be derived by interpolating the values of a selected set of points.

Point-based value iteration methods have been very successful in solving problems which are orders of magnitude larger than classical POMDP problems. The PBVI algorithm [Pineau *et al.*, 2003] performs value updates on a small set $B = \{b_0, b_1, ..., b_m\}$ of reachable points. The error of the approximation is bounded and it can be decreased by expanding the set of beliefs. However, value improvement depends to a large extent on which belief points are added to this set. Hence, the choice of belief points is a crucial problem in point-based value iteration, especially when dealing with large problems. This has been discussed by several researchers. [Spaan and Vlassis, 2005] explored the use of a large set of randomly generated reachable points. [Pineau *et al.*, 2003] suggested several heuristics for sampling reachable belief states. [Smith and Simmons, 2005; 2004] designed an algorithm which maintains an upper and lower bound on the value function to guide the search for more beneficial beliefs.

Generalization over the entire belief space is done based on the assumption that nearby points are more likely to have close values. This assumption is based on the fact that the optimal value function is a piecewise linear and convex function over the continuous belief space. The quality of approximations depends strongly on the chosen beliefs. If it was possible to plan for all the reachable belief points in the desired planning horizon h, then the solution would be exactly optimal. Unfortunately, although the set of reachable belief states from a given initial state is finite in a finite horizon, this set is exponential in the horizon length $((|A||Z|)^h)$ and computing an optimal solution over this set is computationally impossible for reasonable values of h. When the agent cannot afford to consider all the reachable belief states that it might encounter in future, it must make a choice as to how to sample these points. Ideally, we need to sample enough points to build a good approximation but the set should be small enough to allow a quick computation. The performance of point-based algorithms can be improved by eliminating unnecessary reachable points or

concentrating on more important points in the belief space. In this chapter we illustrate the importance of point selection for this class of methods and propose new heuristics that try to solve this problem in a more efficient way within the PBVI framework.

First, we propose and investigate a new method for point selection in belief space called *core belief value iteration (CBVI).* This approach generates a set of linearly independent belief vectors which are encountered by the agent while interacting with the environment, then PBVI is used to find the value of these beliefs and to generalize the value function over the entire belief simplex. We discuss how this approach can improve the worst case error bound of PBVI. Then we make some corrections to the reachability metric proposed by Smith and Simmons [2005], and apply this metric in PBVI as a distance measure to select points in the reachable belief space. We call this heuristic *reachability-based.* The previously existed approaches for point selection in point-based methods sample the points uniformly with respect to their level of reachability form a given initial belief. The reachability metric is designed to give more priority to points that are reachable in the near future. The intuition is to give more weight to points that are reachable in near future when selecting points for backup.

In Section 4 we investigate the effect of using the estimated value of points instead of the distance between them as a selection criterion. In Section 5 we propose and investigate a new strategy, that we call *threshold-based* for point selection in belief space. In this approach we try to balance between choosing points in breadth and in depth in forward search, from a given initial belief. The fundamental idea behind this technique is to give priority to beliefs reachable in the near future while still taking into account the distance between a candidate point and the current point set B. This is motivated by the observation that the complexity of the optimal value function can be inferred, to some extent, from the difference between the number of belief states being backed up, $|B_i|$, and the number of α-vectors representing the current approximate value function, $|\Gamma_i|$. Whenever this difference is large, a lot of points share the same optimal policy. Therefore, we can sample points more sparsely, imposing a larger threshold on their distance to the current set of beliefs. A small (or zero) difference between $|B_i|$ and $|\Gamma_i|$ indicates that for a more accurate approximation

we need to sample more densely form the space of reachable beliefs, because different beliefs have different optimal actions.

We provide empirical results in Section 6 for comparing our approaches with previously suggested techniques, on a set of standard POMDPs. These results suggest that the threshold-based method is the winning approach.

4.1. Remarks on the Point-Based Value Iteration Algorithm

The exact value iteration methods are mainly based on dynamic programming and they exhibit monotonicity and contraction mapping [Bertsekas, 1987]. Monotonicity ensures that the error of the value estimates for all points in belief space is decreased from one iteration to the next. The contraction mapping property guarantees that the value iteration method converges to a unique optimal value function. However, in the PBVI algorithm, from one iteration to the next, there is no guarantee that the value function over the entire belief space is improved, although this property still holds for the current selected set of beliefs B. Also, there is no theoretical guarantee of convergence to the optimal value function for all points of the belief space unless we include them all in B.

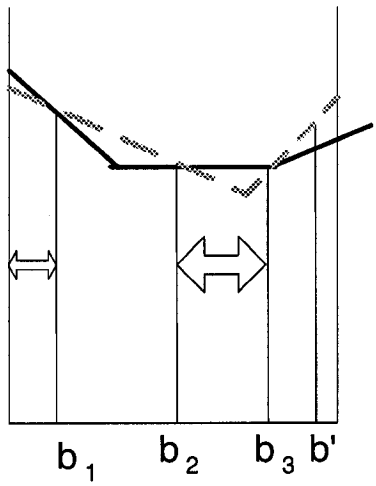


FIGURE 4.1. The value of a point not included in B can be decreased from one expansion to the next. The black line segments show the current value function based on $B = \{b_1, b_2\}$ by adding the point b_3 in the set B the value function can be represented by the gray line segments. In the intervals shown by the arrows the next value estimates will be decreased

In practice, the estimated value of a reachable point $b \notin B$ can be decreased during the iterative progress of the PBVI algorithm. Figure 4.1 illustrates the situations where the value of a point b is decreased. This figure depicts the situation in the belief set expansion phase of PBVI, where adding the point b_3 to B causes the dotted lines to be the only α-vectors that PBVI maintains in representing the value function. However, this leads to a decrease in the value estimate, compared to the previous estimate, for points b in the areas shown by the arrows. For this reason, adding extra points to back up does not necessarily lead to better value estimates over the entire space of reachable beliefs. This is useful in explaining the behavior of the derived policy at execution time, as we will see in Section 6.

Ideally, we would want to reduce the worst-case error of the value function estimate by including the points making this error in B. This implies that in a forward search for reachable belief points from B we should be looking for a parameter that best describes the amount of error in the value estimates for a particular set of candidate points B_c. Formally, this error at a candidate point b' is defined by the difference between the current estimate of the value function V_B at b' and the perspective value estimate while we include b' in B ($V_{B\cup\{b'\}}$):

$$\forall b' \in B_c : error(b') = |V_B(b') - V_{B\cup\{b'\}}(b')|$$

The best candidate points to include in B, in order to avoid the worst case error, are the ones which introduce large errors. Of course, to make a precise estimation of this error we should add all candidate points in the forward search to the set B to quantify $V_{B\cup\{b'\}}$. This requires a huge amount of computation since we need to perform the expensive point-based value backup ($O(|A||Z|^h)$) for $|B_c| = |A||Z||B|$ times. While it might not be efficient to find the exact difference in the value estimates beforehand, the geometrical distance between candidate points and the current B is readily available. Therefore, some approaches consider the geometrical distance between B_c and B as a good (although not the best) parameter to indicate the potential impact on the value function estimates. The fact that the approximate value function is piecewise linear and convex justifies this. Figure 4.2 shows the first expansion of the point set $B = \{b_0\}$ with three reachable points from b_0 as candidate points. The grey line in this figure is the current representation of the value function

and the black hyperplanes represent the optimal value function. It can be seen from this figure that b'_2 introduces the worst error ($err(b'_2)$), as defined in the above equation, while b'_1 is geometrically farthest from b_0. The geometrical distance of b'_1, b'_2, and b'_3 from b_0 is a reflection of the prospective error that these points introduce but this is not a perfect estimate. In this example d_1 and d_3 reflect well the $err(b'_1)$ and $err(b'_3)$) but the choice between b'_1 and b'_2 would be made incorrectly based on d_1 and d_2.

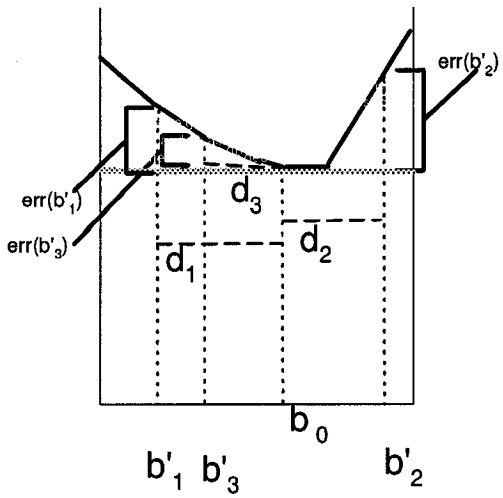


FIGURE 4.2. Candidate points reachable from a given point b_0.

The structure of the underlying problem seems to play the most crucial role in the efficiency of point-based approximation. If $|S|$ is very large but the nature of the domain is such that reachable beliefs are sparsely located in small clouds along some dimensions (low belief entropy), then we expect to get a good approximation using a small set B (see Figure 4.3). The opposite scenario can happen when $|S|$ is not too large but the beliefs are scattered all over the belief simplex. Then it is not easy to cover all by considering a reasonable size B unless the value function is flat. It is very hard to characterize theoretically the relationship between the number of belief points and the approximation error. The smoothness of the expected value function and the slope of its representing hyper-planes could help to find

a set of points which provide a good approximation. This type of quantitative analysis is still an open problem in the POMDP literature.

4.2. Core-Belief Value Iteration

For any belief set B and horizon h, PBVI estimates the finite horizon point-based value function V_B^h. The error between V_B^h and the finite horizon value function over the entire belief space, V^h, denoted by $\varepsilon_h = \|V_B^h - V^h\|_\infty$, was shown to be bounded [Pineau *et al.*, 2003]:

$$\varepsilon_h \leq \frac{(R_{max} - R_{min})}{1-\gamma}\delta_B \tag{4.1}$$

where the parameter δ_B depends on the sampled belief set B and the set of all reachable beliefs Δ (as described in Section 2.3.3). The parameter δ_B is defined as:

$$\delta_B = \max_{b' \in \Delta} \min_{b \in B} \|b - b'\|_1 \tag{4.2}$$

We propose to start directly with the distance δ_B described above and choose the points in such a way that the distance never reaches its maximum, even if we do no expansion of the point set B.

4.2.1. Generating Core Beliefs

In the context of predictive state representations in the previous chapter, we defined the system dynamic matrix, which has a maximal set of linearly independent tests and histories. Each entry of finite dimension corresponds to the probability of a particular test from a particular history. If the POMDP model is given the entries in this matrix, M_{ij}, can be computed as:

$$M_{ij} = b_h^T(s_i)U(q_j)$$

[1]The study of domain condition in the context of convex piecewise linear stochastic programs in [Shapiro *et al.*, 2000] is very inspiring. Similar theoretical analysis can shed some light on a way to find an efficient representative set of points in point-based framework.

where q_j, $U(q_j)$ are defined as in Section 3.1.2. Linearly independent histories consequently generate linearly independent reachable belief states. The set of linearly independent tests and histories can also be constructed if the SDM is acquired from data. Algorithm 5 presents an approach to determine the linearly independent histories and tests. Starting from all one-step tests and histories, the algorithm proceeds by computing elements of the SDM and extracting a maximum set of linearly independent rows and columns (histories and tests). Then it repeatedly computes one-step extensions to all the elements in these sets until it finds a maximal set. This algorithm generates a set of linearly independent histories Q_H, whose elements can be used to produce a set of reachable and linearly independent belief states. We call them *core-beliefs* and denote the set by CB. Belief points in CB span the space of reachable beliefs (i.e. can be linearly combined to produce any reachable belief state).

Algorithm 5 Core Belief Discovery

$i \leftarrow 0$
$Rank(i) \leftarrow 0$
$Q_H \leftarrow \{\}$; $Q_T \leftarrow \{\}$
repeat
  $i \leftarrow i+1$
  $\mathcal{H}_i \leftarrow$ all one-step extensions of Q_H
  $\mathcal{T}_i \leftarrow$ all one-step extensions of Q_T
  construct SDM_i from $\mathcal{H}_i \cup Q_H$, $\mathcal{T}_i \cup Q_T$
  $Rank(i) \leftarrow$ rank of SDM_i
  $Q_H \leftarrow$ histories for linear independent rows in SDM_i
  $Q_T \leftarrow$ tests for linearly independent columns in SDM_i
until $Rank(i) = Rank(i-1)$
Return Q_H

Note that finding the rank of a matrix (each matrix expansion of Algorithm 5) runs in $O(n^3)$ where n is the number of rows. But there are $O(|A||Z||Q_{H_i}|)$ rows in SDM_i, where $|A|$is the size of the action set, $|Z|$ is the size of the observation set, and Q_{H_i} is the total number of core tests of length less than or equal to i. The length of these tests is bounded by the size of the state space $|S|$. For a POMDPs with large number of actions and observations, extending tests by all one-step combinations of actions and observations, can make the matrix huge. In practice, memory will be an implementation issue for these cases.

To avoid this problem, we used some heuristics for extending tests and histories instead of extending by all actions and observations to solve this problem. A detailed discussion is given in the experimental results in Section 7.

The CBVI algorithm starts with a small initial set of belief points, CB, where $|CB| \leq |S|$. It applies the first series of backup operations, then the set is grown as in PBVI until a satisfactory solution is obtained.

4.2.2. Error Bound

The theoretical performance analysis of CBVI focuses on the PBVI error (4.1). The norm-1 metric (total deviation distance) and the property that belief points are probability vectors ($||b||_1 = 1$) ensure that $\delta_B \leq 2$.

THEOREM 4. *Including of the core belief set CB in the point set B ensures that the distance* $\delta_B < 2$.

PROOF. Each belief state $b' \in \Delta$ is a linear combination of points in CB, therefore:

$$b'(i) = \sum_{b_j \in CB} c_j b_j(i) \tag{4.3}$$

Therefore:

$$||b' - b||_1 = \sum_i \sum_{b_j \in CB} |c_j b_j(i) - b_j(i)| < ||b'||_1 + ||b||_1 = 2 \tag{4.4}$$

□

It is worth noting that the loose error bound described in equation (4.1) does not really reflects the strength and the precision of the point-based approximation in general. Since even in a hypothetically worst case the difference between the values of two belief points would be $\frac{(R_{max} - R_{min})}{1-\gamma}$, although the distance of the two points may be 2. This is true for the case of core-beliefs as well. The fact that δ_B never reaches the maximum value if CB is included in the set B, does not properly reflect the real effect of these points in the precision of the point-based approximation in practice. Another thing to notice is that including core-beliefs in B does not mean that the error will converge to zero. Therefore, we still need to add points to B in order to improve the solution quality. The hope is that if $CB \subset B$ we

can reach a better set of points faster. However, this will depend on the structure of the domain. A simple example to demonstrate this is presented in Figure 4.3. In the left side of this figure, core beliefs can be good representatives of the reachable belief simplex if the value function is smooth and has close values for points in each cluster. Another interesting point in this case is that if the beliefs in one of the clouds are rarely reached then they can be overlooked by a uniform sampling of reachable points. However, this is less likely to happen in CBVI.

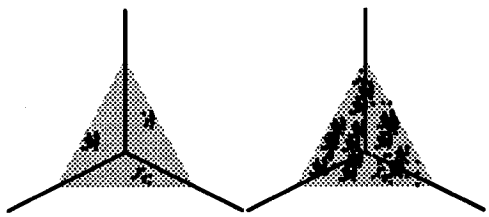


FIGURE 4.3. The distribution of reachable belief states make the effect of CBVI more pronounced at the left and less at the right.

The right side of this figure illustrates the situation in which reachable beliefs are distributed uniformly and densely all over the belief space. In this case a set of core beliefs does not necessarily provide a better solution than almost any random sample of reachable beliefs of the same size. However, this is a very unlikely situation as most of the POMDP application domains exhibit a great deal of structure in their state space and their dynamics, which makes the reachable belief space considerably smaller and more structured than a uniform distribution of beliefs.

4.3. Choosing Belief Points Using Reachability Analysis

In order to reason about the space of reachable beliefs, one can consider the initial belief vector, b_0, and then all possible one-step sequences of actions and observations following it. By considering all one-step action-observation sequences that can occur from these beliefs, we can obtain all beliefs reachable from b_0 in two steps, and so on. This will produce a tree rooted at the initial belief state b_0. The space of reachable beliefs consists of all the nodes of this tree (which is infinite in general, but cut to a finite depth in finite-horizon tasks).

The *discounted reachability* ρ is a mapping from the space of reachable beliefs Δ to the real numbers, defined as: $\rho(b) = \gamma^L$ where L is the length of the shortest sequence of transitions from the initial belief state b_0 to b and γ is the discount factor given in the POMDP model. This definition implies that for all beliefs b'_{baz},

$$\rho(b'_{baz}) \geq \gamma\rho(b) \tag{4.5}$$

because either b'_{baz} has been obtained in one step from b (in this case we have equality in the above equation), or it has been obtained somewhere earlier (along a shorter path) from b_0. Based on the definition of discounted reachability, Smith and Simmons [2005] define a generalized sample spacing measure δ_P $(0 \leq p < 1)$. Their argument is that they want to give more weight to beliefs that are reachable in the near future, because their values influence the value function estimates more. To do this, they divide the L_1 norm of the beliefs by $(\rho(b))^p$. However, this division actually has an opposite effect, emphasizing more beliefs that are in the distant future. To correct this problem, we redefine the sample spacing measure in [Smith and Simmons, 2005] as:

$$\delta_p(b) = \max_{b\in\Delta} \min_{b'\in\beta} \|b - b'\|_1 [\rho(b)]^p \tag{4.6}$$

where p is a parameter in $[0, 1)$. Note that the error defined by δ_B in [Pineau *et al.*, 2003](Equation (4.2)) is a special case of $\delta_p(b)$ with $p = 0$, where a uniform weight is assigned to each reachable belief. We now show that the results in [Smith and Simmons, 2005] hold with this new definition of $\delta_p(b)$. First we need to show that the update operator applied to points selected by the above metric is a contraction mapping. To do this, consider the following weighted norm:

$$\|V - \bar{V}\|_\rho = \max_b |V(b) - \bar{V}(b)|\rho(b)^p \tag{4.7}$$

In other words, this is like a max norm but the elements are weighted by weights ξ.

THEOREM 5. *The exact Bellman update in equation (2.8) is a contraction mapping under the norm defined in equation (4.7) with contraction factor* γ^{1-p}.

PROOF. For the proof, it is easier to consider action-value functions. We will use the same notation as Smith and Simmons [2005], in order to facilitate the comparison with their results. Let $Q_a^V(b)$ be the value of executing action a in belief state b, given that the value function estimate for states is V:

$$Q_a^V(b) = R(b,a) + \gamma \sum_{b'} Pr(b'|b,a)V(b')$$

For any action $a \in A$, and for any value function estimators V and $\bar{V}$ we have:

$$\begin{aligned}
\|Q_a^V - Q_a^{\bar{V}}\|_{\rho^p} &= \max_b |Q_a^V(b) - Q_a^{\bar{V}}(b)|[\rho(b)]^p \\
&= \max_b \gamma \sum_{b'} P(b'|b,a)|V(b') - \bar{V}(b')|[\rho(b)]^p \\
&= \max_b \gamma \sum_{b'} P(b'|b,a)|V(b') - \bar{V}(b')| \left[\frac{\gamma\rho(b)}{\gamma}\right]^p \\
&\leq \max_b \gamma \sum_{b'} P(b'|b,a)|V(b') - \bar{V}(b')| \left[\gamma^{-1}\rho(b')\right]^p \text{ (using (4.5))} \\
&\leq \max_b \gamma^{1-p} \sum_{b'} P(b'|b,a) \max_{b''} |V(b'') - \bar{V}(b'')| \left[\rho(b'')\right]^p \\
&= \gamma^{1-p} \sum_{b'} P(b'|b,a)\|V - \bar{V}\|_{\rho^p} \\
&= \gamma^{1-p}\|V - \bar{V}\|_{\rho^p}
\end{aligned}$$

As a side note, we need to mention that equation (10) in [Smith and Simmons, 2005]:

$$\|Q_a^V - Q_a^{\bar{V}}\|_{\rho^p} \leq \max_b \gamma \sum_{b'} Pr(b'|b,a) \frac{|V(b') - \bar{V}(b')|}{\gamma\rho(b')^p}$$

will be violated by their definition of $\delta_p(b)$ as: $\delta_p(b) = \max_{b \in \bar{A}} \min_{b' \in B} \frac{\|b-b'\|_1}{[\rho(b)]^p}$. Also the contraction factor as stated there is not consequently correct. Let HV be a "greedification" operator on the value function, defined as:

$$HV(b) = \max_a Q_a^V(b), \forall b$$

Then, for any belief state b in the reachable belief space Δ we have:

$$|HV(b) - H\bar{V}(b)| \leq \max_a |Q_a^V(b) - Q_a^{\bar{V}}(b)|$$

Multiplying both sides by $[\rho(b)]^p$ we obtain:

$$|HV(b) - H\bar{V}(b)|[\rho(b)]^p \leq \max_a |Q_a^V(b) - Q_a^{\bar{V}}(b)|[\rho(b)]^p$$

Maximizing over b, we obtain:

$$\|HV - H\bar{V}\|_{\rho^p} \leq \max_a \|Q_a^V - Q_a^{\bar{V}}\|_{\rho^p} \leq \gamma^{1-p}\|V - \bar{V}\|_{\rho^p}$$

This completes the proof. □

The next theorem bounds the error of a policy based on an approximate value function $\hat{V}$.

THEOREM 6. *The expected error introduced by a look ahead policy $\hat{\pi}$ induced by an approximate value function $\hat{V}$, starting at the initial belief b_0 is bounded by:*

$$\varepsilon_{\rho^p}^{\hat{\pi}} \leq \frac{2\gamma^{1-p}}{1-\gamma^{1-p}}\|V^* - \hat{V}\|_{\rho^p}$$

PROOF. Let $b \in \Delta$ be an arbitrary belief state and π^* be the optimal policy. Let $V^{\hat{\pi}}$ be the value function of policy $\hat{\pi}$. Note that $Q_{\hat{\pi}(b)}^{V^{\hat{\pi}}}(b) = V^{\hat{\pi}}(b)$. Since $\hat{\pi}$ is a lookahead policy induced by $\hat{V}$ then $Q_{\hat{\pi}(b)}^{\hat{V}}(b) = \max_a Q_a^{\hat{V}}(b) = H\hat{V}(b)$. The optimal value function is the fixed point of the Bellman equation, therefore $V^* = HV^*$. We have:

$$\begin{aligned}
|V^{\pi^*}(b) - V^{\hat{\pi}}(b)| &= |V^*(b) - Q_{\hat{\pi}(b)}^{V^{\hat{\pi}}}(b)| \\
&= |V^*(b) - Q_{\hat{\pi}(b)}^{V^{\hat{\pi}}}(b) + Q_{\hat{\pi}(b)}^{\hat{V}}(b) - Q_{\hat{\pi}(b)}^{\hat{V}}(b)| \\
&\leq |V^*(b) - H\hat{V}(b)| + |Q_{\hat{\pi}(b)}^{\hat{V}}(b) - Q_{\hat{\pi}(b)}^{V^{\hat{\pi}}}(b)| \text{ (by grouping terms)} \\
&\leq |HV^*(b) - H\hat{V}(b)| + \gamma \sum_{b'} Pr(b'|b, \hat{\pi}(b))|\hat{V}(b') - V^{\hat{\pi}}(b')|
\end{aligned}$$

Multiplying both sides by $[\rho(b)]^p$ we get:

$$\begin{aligned}
|V^*(b) - V^{\hat{\pi}}(b)|[\rho(b)]^p &\leq |HV^*(b) - H\hat{V}(b)|[\rho(b)]^p \\
&+ \gamma \sum_{b'} Pr(b'|b, \hat{\pi}(b))|\hat{V}(b') - V^{\hat{\pi}}(b')| \left\lceil \frac{\rho(b)\gamma}{\gamma} \right\rceil^p \\
&\leq |HV^*(b) - H\hat{V}(b)|[\rho(b)]^p \\
&+ \gamma^{1-p} \sum_{b'} Pr(b'|b, \hat{\pi}(b))|\hat{V}(b') - V^{\hat{\pi}}(b')|[\rho(b')]^p \\
&\leq |HV^*(b) - H\hat{V}(b)|[\rho(b)]^p + \gamma^{1-p}\|\hat{V} - V^{\hat{\pi}}\|_{\rho^p}
\end{aligned}$$

By taking a max with respect to b from both sides and the definition of the norm in Equation (4.7) we obtain:

$$\begin{aligned}
\|V^{\pi^*} - V^{\hat{\pi}}\|_{\rho^p} &\leq \|HV^* - H\hat{V}\|_{\rho^p} + \gamma^{1-p}\|\hat{V} - V^{\hat{\pi}}\|_{\rho^p} \\
&\leq \gamma^{1-p}\left(\|V^* - \hat{V}\|_{\rho^p} + \|\hat{V} - V^{\hat{\pi}}\|_{\rho^p}\right) \text{ (using theorem 4)} \\
&= \gamma^{1-p}\left(\|V^* - \hat{V}\|_{\rho^p} + \|\hat{V} - V^* + V^* - V^{\hat{\pi}}\|_{\rho^p}\right) \\
&\leq \gamma^{1-p}\left(\|V^* - \hat{V}\|_{\rho^p} + \|\hat{V} - V^*\|_{\rho^p} + \|V^* - V^{\hat{\pi}}\|_{\rho^p}\right) \\
&\leq \gamma^{1-p}\left(2\|V^* - \hat{V}\|_{\rho^p} + \|V^* - V^{\hat{\pi}}\|_{\rho^p}\right)
\end{aligned}$$

Solving the above equation we obtain: $\|V^* - V^{\hat{\pi}}\|_{\rho^p} \leq \frac{2\gamma^{1-p}}{1-\gamma^{1-p}}\|V^* - \hat{V}\|_{\rho^p}$. For the initial state $\rho(b_0) = 1$, hence the regret of $\hat{\pi}$ starting at b_0 will be bounded as follows:

$$V^*(b_0) - V^{\hat{\pi}}(b_0) \leq \frac{2\gamma^{1-p}}{1-\gamma^{1-p}}\|V^* - \hat{V}\|_{\rho^p}$$

□

THEOREM 7. *Let H_B be the update operator applied using only beliefs from set B. Then the error induced by a single application of H_B instead of the true operator H is bounded as:* $\|HV - H_B V\|_{\rho^p} \leq \frac{(R_{max} - R_{min})\delta_p(B)}{1-\gamma^{1-p}}$

where R_{max} and R_{min} are the maximum and minimum rewards that can be achieved, in the POMDP model.

PROOF. We follow a similar argument to the one in [Pineau *et al.*, 2003]. Let b' be the reachable belief that is currently not included in the set B and has the worst error in ρ^p norm. Let $b \in B$ be the belief in the current belief set B whose α-vector, α, is currently the best vector for b'. Suppose the true optimal α-vector at b' would be α', but instead we use the estimate α that comes from b. Therefore, we have:

$$\begin{aligned}
\|HV - H_B V\|_{\rho^p} &\leq (\alpha' b' - \alpha b')\rho(b') = (\alpha' b' - \alpha' b + \alpha' b - \alpha b')\rho(b') \\
&\leq \left[\alpha'(b' - b) + \alpha(b - b')\right][\rho(b)]^p \text{ (because } \alpha \text{ is the optimal vector at } b) \\
&\leq |(\alpha' - \alpha)(b' - b)|[\rho(b')]^p \\
&\leq \|(\alpha' - \alpha)\|_\infty |(b' - b)|_1 [\rho(b')]^p \text{ (Holder inequality)} \\
&\leq \|\alpha' - \alpha\|_\infty \max_{b'} \min_{b} \|b' - b\|_1 [\rho(b')]^p \\
&= \|\alpha' - \alpha\|_\infty \delta_p(B) \leq \frac{R_{max} - R_{min}}{1 - \gamma^{1-p}} \delta_p(B)
\end{aligned}$$

□

We used result for the contraction factor in the denominator.

THEOREM 8. *The accumulated error* $\|V_t^B - V_t^*\|_{\rho^p}$ *at any update step t, is at most* $\frac{(R_{max} - R_{min})\delta_p(B)}{(1-\gamma^{1-p})^2}$

PROOF. The argument is analogous to Theorem 1 of [Pineau *et al.*, 2003]:

$$\begin{aligned}
\|V_t^B - V_t^*\|_{\rho^p} &= \|H_B V_{t-1}^B - H V_{t-1}^*\|_{\rho^p} \text{ (fixed point)} \\
&\leq \|H_B V_{t-1}^B - H V_{t-1}^B\|_{\rho^p} + \|H V_{t-1}^B - H V_{t-1}^*\|_{\rho^p} \\
&\leq \varepsilon_{\rho^p}^{\pi_t} + \|H V_{t-1}^B - H V_{t-1}^*\|_{\rho^p} \text{ (definition of the error)} \\
&\leq \varepsilon_{\rho^p}^{\pi_t} + (\gamma^{1-p} \|V_{t-1}^B - V_{t-1}^*\|_{\rho^p} \text{ (contraction)} \\
&= \varepsilon_{\rho^p}^{\pi_t} + \gamma^{1-p} \varepsilon_{\rho^p}^{\pi_{t-1}} \text{(definition of the error)} \\
&\leq \frac{(R_{max} - R_{min})\delta_p(B)}{(1 - \gamma^{1-p})} + \gamma^{1-p} \varepsilon_{\rho^p}^{\pi_{t-1}} \text{ (concluded from the above theorem)} \\
&\leq \frac{(R_{max} - R_{min})\delta_p(B)}{(1 - \gamma^{1-p})^2} \text{ (series sum)}
\end{aligned}$$

□

Algorithm 6 Average-norm Belief Expansion (Initial belief set B)

for all $b \in B$ **do**
  for all $a \in A$ **do**
    Sample the current state s from b
    Sample the next state s' from $T(s,a,\cdot)$
    Sample the next observation z from $O(a,s',\cdot)$
    Compute the next belief b'_{baz} reachable from b
  end for
  $b^* = \arg\max_{b'_{baz}} \delta(b'_{baz})$
  $B = B \bigcup \{b^*\}$
end for
Return B

Algorithm 6 presents an approach for expanding the belief set using the reachability heuristic. Note that instead of looking at all reachable beliefs, we just sample one possible successor for each belief in the current set. Of course, this algorithm could be changed to take more samples, or to look farther into the future. However, farther lookahead is less likely to matter, because of the weighting used in the heuristic. The drawback of this approach is that after a few cycles, the strategy will sample points from a rather restricted set and that could lead to smaller and smaller improvements. It must be noted that, for instance, for domains with deterministic observations and transitions, this approach gives very similar results to the stochastic simulation with explorative actions (SSEA) heuristic [Pineau *et al.*, 2003], because of the narrow distribution of reachable beliefs. Considering that PBVI converges to a good approximation in just a few expansions, and that the factor γ is usually between 0.75 to 0.99, the effect of $\delta_P(B)$ is not much different in Algorithm 6 compared to the effect of $\varepsilon(B)$ in SSEA-PBVI.

4.3.1. Breadth First Belief Selection

A more radical approach for emphasizing belief states that are reachable in the near future is to include them all into the set B. Theoretically, this should provide the best approximation in terms of the weighted norm that we are considering. But in many problems this is not feasible, because the size of the fringe of the belief tree grows exponentially in

the horizon length. But, in order to get an estimate of how well we could do in this case, we consider adding one belief point for every possible action from every belief in the current set. The observations are still sampled. The main idea is that we typically expect the number of actions to be small, but the number of possible observations to be quite large. Obviously, this could be extended to sample k observations for each action. This idea is expressed in Algorithm 7. In this case, the size of B for the next set of point-based backups will be increased by at most $|B||A|$ in each expansion. Because this approach adds significantly more beliefs at each step, we would expect it to obtain a good approximation in a smaller number of expansions. But we want to ensure that the number of beliefs is also small enough.

Algorithm 7 Breadth First Belief Expansion

for all $b \in B$ **do**
  for all $a \in A$ **do**
    Sample s from b
    Sample s' from $T(s,a,.)$
    Sample z from $O(a,s',.)$
    Compute the next belief $b_{new} = b'_{b,a,z}$ reachable from b
    If b_{new} is not already in B
    $B = B \bigcup \{b_{new}\}$
  end for
end for
Return B

4.4. Choosing Points Using Current Value Function Estimate

The methods presented so far have not directly taken advantage of the anytime feature of the PBVI algorithm to select points for the next cycle. The two heuristics considered in the previous sections try to find a small set of essential points that can be used to represent the value function; however, these are only heuristics, do not always work well as we discussed earlier in this chapter.

We suggest and investigate a value-based method which samples new points using the most recent value function. It attempts to include in the set of selected beliefs points that

are critical for getting better value estimates. As we discussed in the Section 4.1, the generalization of the current point-based value function to beliefs not included in B can have the danger of overestimation of the value of some points and underestimation of some others. We build upon the fact that reachable states with highest and lowest expected values can be desirable points for a better precision of the value function. Unfortunately we do not have the real value function to determine these points correctly. We use the current value function to identify those potential points and include them in B to find a better approximation. Algorithm 8 presents this selection method. As shown in this algorithm, we add at most $2|B|$ points to the current belief set and the size of the set B therefore will become at most tripled after each expansion. The Point-based Error Minimization Algorithm

Algorithm 8 Value-based Belief Expansion

for all $b \in B$ **do**
  for all $a \in A$ **do**
    Sample s from b
    Sample s' from $T(s,a,.)$
    Sample z from $O(a,s',.)$
    Compute the next belief b'_{baz} reachable from b
  end for
  $b^{max}_{new} =$ arg max $_{b_a} b_a \dot{\alpha}$
  $b^{min}_{new} =$ arg min $_{b_a} b_a \dot{\alpha}$
  $B = B \bigcup \{b^{min}_{new}, b^{max}_{new}\}$ (if they are not already in B)
end for
Return B

(PEMA), introduced recently by Pineau and Gordon [2005], uses a greedy error reduction heuristic aimed to quantify the error w.r.t the current value function. This approach seems to perform similarly to PERSEUS [Spaan and Vlassis, 2005]. Although, there is still no formal proof on the monotonic improvement of the value function in PEMA.

4.5. Threshold-Based Belief Selection

We have observed that the beliefs reachable in one time step are not necessarily the best candidates to improve the value function estimate, and that including all of these beliefs in the point set B results in many unnecessary value updates or backups. Theoretically, a

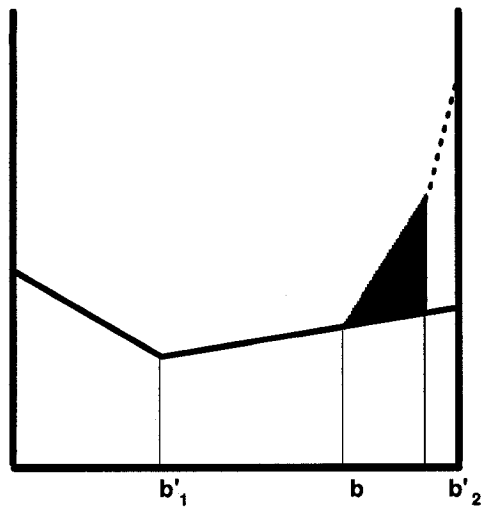


FIGURE 4.4. Geometric interpretation of error introduced while selecting points based on the maximum distance in forward simulation.

subset of these points which has a larger distance (e.g. in norm-1 sense) to the current set B has more potential for improvement on the current approximation due to the basic intuition that nearby points have nearby values. However, considering only the most distant points to B is not always the best criterion to reduce the approximation error either (as we discussed in 4.1). It seems that a good sampling strategy should consider both the reachability distance and the geometric distance between points. An example of this type was examined through the core belief heuristic. Core beliefs are the scattered along the intrinsic dimensions of the reachable belief space and at the same time they are reachable in different time steps (through different levels of a forward search).

In this section we try to find another type of middle ground between beliefs reachable in breadth and depth. We consider adding all candidate belief points, b^c, for every possible action $a \in A$ from every belief $b \in B$, such that their *norm*-1 distance from the current B is at least e.

$$d(b^c,B) = \min_{b \in B} ||b - b^c||_1 > e \tag{4.8}$$

Algorithm 9 presents this idea. In this case the size of B for the next set of point-based backups will be increased more than in the case of the previously used methods, which at most double the size of B. But if this approach adds more significant beliefs at each

step, we would expect it to obtain a good approximation in a smaller number of expansions. Therefore, the threshold e can be related to two parameters: 1) the stochasticity of

Algorithm 9 Threshold-Based Belief Expansion (Initial belief set B, threshold e)

```
for all b ∈ B do
  for all a ∈ A do
    Sample current state s from b
    Sample next state s' from T(s,a,·)
    Sample next observations z from O(a,s',·)
    Compute the next belief b'_baz reachable from b
    If d(b'_baz, B) > e then B = B ∪ {b'_baz}
  end for
end for
Return B
```

the domain (i.e. degree of stochasticity in action transitions and observation emissions) which indicates how dense or sparse the space of reachable beliefs is expected to be; 2) the expected complexity of the value function that is captured in part by the size of the current optimal policy. While (1) can be determined having the model of the world, it is not obvious how to measure (2) before computing the optimal value function. The possible values of e are in the interval $[0,2]$. Note that the PBVI-SSEA heuristic [Pineau *et al.*, 2003] is a conservative version of the threshold-based expansion, which can overlook critical points in each expansion by considering only the most distant point from the current B, $e = \max_{b'} d(b',B)$. The $e = 2$ threshold does not let the set B to expand at all. Although the error bound on PBVI (equation (4.2)) is quite loose, it still holds with only a single point $B = \{b_0\}$. As we decrease e, naturally, the PBVI error should be decreased and lead to a tighter bound. Therefore, the breadth first heuristic (selecting all points one-step away from b), which can be considered as a special case of the above algorithm with $e = 0$, takes the approximation error of PBVI down to zero. However, as we decrease e, the size of B can grow dramatically (depending on the problem domain) and make it hard to perform point-based value update on these points. There is still no theoretical characterization of domains to suggest the best values of e or that quantifies the PBVI error reduction by introducing a tighter bound. Although the point selection method in Algorithm 9 still does not make the

decrease in error monotonic from one expansion to the next, it attempts to keep ε_B as low as e at all times.

4.6. Discussion of Related Work

While point-based techniques are still constrained by the curse of dimensionality, other approaches have been proposed that try to use a set of belief points to obtain a low dimensional representation first, and then solve the reduced form by other existing methods. The exponential family Principal Components Analysis (EPCA) algorithm of Roy and Gordon [2003] is an example of this type of approach. It finds a representation of the beliefs in a exponentially lower dimension by taking advantage of a particular kind of belief space structure. This algorithm uses the low-dimensional features to convert the POMDP into a tractable MDP and then uses value iteration methods to find the solution. Whenever the belief space is very sparse and most plausible belief points are embedded in clusters of low-dimension, EPCA provides a fast and good approximation. Although this approach can potentially solve large problems, it provides no guarantees on the quality of solutions in the general case. Our CBVI approach described in this chapter also benefits from the low-dimensional embedding of reachable belief states similar to EPCA, although the reduced dimensionality used by CBVI is linear and way too small compare to EPCA.

The Value Directed Compression with Bounded Policy Iteration (VDCBPI) algorithm by Poupart and Boutilier [2004b] is another approximation technique for POMDPs which tries to overcome the curse of dimensionality and the curse of history at the same time. This algorithm combines the value-directed compression technique [Poupart and Boutilier, 2000] for targeting the curse of dimensionality with the bounded policy iteration (BPI) [Poupart and Boutilier, 2004a] for overcoming the curse of history. The BPI algorithm incrementally constructs a finite state controller using policy iteration and applies a backup operator called *bounded backup*. Similar to point-based methods, this approach keeps the size of the controller limited. However, unlike the point-based backup operator, the bounded backup guarantees value improvement for all belief states as it proceeds. However, the algorithm may get trapped in local optima. It is necessary to mention that the size

of the controllers and therefore the time to get an approximate solutions in BPI (according to the results reported in [Poupart and Boutilier, 2004a]) is significantly larger than PBVI. This confirms that BPI converges slower than PBVI (if not trapped in a local optimum). The VDC part of the VDCBPI algorithm is a dimensionality reduction technique which focuses on belief compression to provide a lower dimensional set of reachable belief points. Like EPCA, this technique tries to capture regularities in the environment. However, the compression that the lossless VDC provides is linear, not exponential like in EPCA. he major advantage of this approach over the EPCA is that it only distinguishes between beliefs that have different value and therefore provides a good approximation in general. A distinctive feature of the lossless VDC method however, is that it uses the smallest subspace that contains the immediate reward vector and is closed under a set of linear functions defined by the state transition and observation model. This is similar to the process of generating core beliefs. The main computational burden in both processes is computing the rank of huge matrixes. The core belief idea of CBVI can realize much of the same compression as lossless VDC. However, CBVI finds the smallest subspace of beliefs that is closed under a set of linear functions on the state transitions and observation model. In Chapter 5 we get back to the VDC algorithm and its relationship to linear PSRs. Using linear programming techniques in VDC and combining with BPI, the authors was able to solve a POMDP with millions of states; however, the number of observations in this domain was very small.

4.7. Empirical Results

This section presents the results of experiments conducted on several POMDP benchmarks used previously in the literature. We compare the performance of the belief selection methods discussed in this chapter, and test the effectiveness of each of these approaches when used in conjunction with PBVI algorithm.

4.7.1. Experimental Setup

The study has been carried out on a set of small domains (in which the size of the state space $|S|$ is less than 20) and a set of large domains. As mentioned earlier, generating

TABLE 4.1. Domains used in the experiments

Domain	$\|S\|$	$\|A\|$	$\|Z\|$	R
4x4-grid	16	4	2	[0, 1]
4x3-grid	11	4	6	[0, 1]
cheese	11	4	7	[0, 1]
network	7	2	4	[-40, 80]
maze33	36	5	17	[-1, 1]
coffee	32	2	3	[-4, 1.5]
hallway	60	5	21	[0, 1]
hallway2	90	5	17	[0, 1]
rocksample[4,4]	257	9	2	[-100, 10]
tag	870	5	30	[-10, 10]

core-beliefs to be used in CBVI involves some complexity issues for large problem domains. Therefore, we have tested the core-belief discovery presented in Algorithm 5 only in small domains. These domains also serve to illustrate a few interesting points in comparing the performance of PBVI with the optimal solutions found by exact algorithms. The small domains used are : 4x4 grid, 4x3 grid, cheese-maze, and network. Larger domains consists of : maze33, hallway, hallway2, robot-coffee, RockSample[4,4], and tag. The full description of the robot-coffee problem can be find in [Poupart and Boutilier, 2003b]. The RockSample problem was defined in [Smith and Simmons, 2004]. The tag problem was defined in [Pineau *et al.*, 2003] and the rest of the domains can be found in Tony Cassandra's repository [Cassandra, 1995]. Table 4.1 summarizes the number of states, actions, observations and the range of rewards in each domain. The evaluation is based on running a set of 250 trajectories, each of length 300 steps starting from a fixed, given initial belief state and following the approximately optimal policy generated by each method. The return of a run is the average return of the trajectories in the run, where the return of each trajectory is the discounted sum of rewards. All the results presented in the tables and in the graphs as *Return* are averages over 10 such independent runs. We compare all results with results we obtained from running the standard PBVI algorithm in which Stochastic Simulation with Explorative Action (SSEA) has been used for belief set expansion. SSEA seems to provide better solutions compare to other heuristics for point selection in PBVI [Pineau *et al.*, 2003].

TABLE 4.2. Empirical results for PBVI and CBVI in the small domains with complete set of core beliefs.

Domain	PBVI	CBVI	PBVI-expand	CBVI-expand	Witness
4x4 grid	3.64 ± 0.09	3.72 ± 0.01	-	-	3.73
4x3 grid	1.81 ± 0.1	1.89 ± 0.05	-	-	1.9
cheese	3.43 ± 0.07	3.45 ± 0.03	-	-	3.48
network	37.44 ± 16.1	19.52 ± 11.1	240.26 ± 5.5	243.92 ± 2.21	244

4.7.2. Empirical evaluation of CBVI Method

In the first set of experiments we wanted to see the effect of having core beliefs in the point set B. Therefore, we initialized the set B with core beliefs for CBVI and with the same number of beliefs, generated by the SSEA heuristic, for PBVI. We first tested both PBVI and CBVI with a number of beliefs equal to the total number of core tests. The results are presented in Table 4.2. For these problems, even with only a few points, we were able to match the solution with the one found by the witness exact solution method. For example for the 4x4 domain, with only one-step core tests in the set B we reached the optimal solution almost always. The total number of core-beliefs for the network domain is still too small to provide a good approximation to the optimal value function. We conjecture that the reason for this result is that the value function is very complex and consists of many α-vectors. Therefore, we kept expanding the belief point sets in both approaches until they converge within a reasonable bound to the optimal solution found with the witness algorithm (the last column in the table). It is necessary to mention that the two approaches do not converge with the same speed in this case. CBVI reaches the value presented after two expansions of B using 25 points in its point set, whereas PBVI needs 4 expansions of its B which used 80 points to get a near optimal solution. Columns PBVI-expand and CBVI-expand of the table present this case for PBVI and CBVI respectively. The other problems converge to the optimal solution with a set B of size equal to the total number of core tests, which was the size of state space in the related POMDP problem. However, as we add more and more points to B the approximation becomes more precise and within a tighter bound around the optimal solution. We note that an initial point set which includes core-beliefs has a smaller standard deviation in the performance. It is not clear if this is due to the

quality of the belief set, or just to using deterministic instead of stochastic initialization. In total, reasonable improvements in solution quality for all the domains have been observed.

In the larger domains (hallway, hallway2 and maze33), computing the complete set of core beliefs is not feasible since there are many reachable beliefs to consider (see our discussion in the section 4.2.1). Therefore, in order to get as many core beliefs as possible in a short time, each iteration of the core-belief discovery algorithm was interleaved with point-based value backups for generating a policy. We used an ε-greedy approach, choosing an action based on the current PBVI estimate of optimal policy. Then for the selected action, we only considered the most likely observation to expand the tests and histories for the next iteration of core-belief discovery. The ε-greedy heuristic was chosen because it actually focuses the computation on good actions that are likely to be used in the optimal policy and which perhaps create more valuable belief points. Table 4.3 shows the result of this experiment on the larger domains. Each expansion indicates an increment to the length of the core-tests corresponding to these core-beliefs. Another way to look at these expansions is that they add beliefs reachable in one more time step to the set B.

As can be seen from these results the CBVI algorithm yields a significant improvement over the PBVI-SSEA heuristic in the hallway and hallway2 problems with respect to the quality of the solution found. The two approaches move at a different pace towards the optimal solution. For the case of Maze33 domain we can see considerable improvement versus PBVI in later iterations. However, core-beliefs of small length (i.e. points reachable in the near future) do not seem to be important in providing good value function estimates. We conjecture that including the complete set of core-beliefs may be more beneficial in this case.

We do emphasize that CBVI has an expensive initial phase for computing the core beliefs. However, the ε-greedy heuristic for generating core beliefs reduces the computational burden significantly, as only one action and one observation are considered in each expansion step from each belief. Once this set is computed, the cost of running PBVI with this initial set is not larger than that of running PBVI. It is necessary to note that the running times of these algorithms depends on the size of the point set B and the size of their value

TABLE 4.3. Empirical results for PBVI and CBVI gradual performance in larger domains.

CoreTestsDiscovery	Method	Maze33	Hallway	Hallway2
Expansion-1				
	PBVI			
	beliefs	9	18	22
	reward	0	0.21 ±0.08	0.15±0.09
	CBVI			
	beliefs	9	18	22
	reward	0	0.50 ±0.02	0.33±0.03
Expansion-2				
	PBVI			
	beliefs	11	22	27
	reward	0	0.25±0.1	0.18±0.07
	CBVI			
	beliefs	11	22	27
	reward	0	0.49 ±0.03	0.34±0.03
Expansion-3				
	PBVI			
	beliefs	15	27	36
	reward	0	0.27 ±0.1	0.23 ±0.09
	CBVI			
	beliefs	15	27	36
	reward	0.23 ±0.2	0.52 ±0.01	0.35 ±0.02
Expansion-4				
	PBVI			
	beliefs	24	31	41
	reward	0.03 ±0.2	0.27±0.09	0.27±0.07
	CBVI			
	beliefs	24	31	41
	reward	1.73 ±0.4	0.52 ±0.01	0.37 ±0.02

function representation (i.e. the number of the α-vectors they use to represent the value function). The number of the α-vectors itself is bounded by the size of B, as these methods maintain at most one α-vector per a belief point $b \in B$.

Overall, these results are very encouraging. However, the current method for generating core-beliefs is not very efficient. There are also other factors which affect the structure of the reachable beliefs and consequently have impact on sampling from this space.

4.7.3. Empirical Evaluations of Reachability-Based, Value-Based, and Threshold-Based Methods

This section includes experiments with the three heuristics described in sections 4.3, 4.4, and 4.5. In this set of experiments, instead of initializing B with pre-computed beliefs, we initialize B with only the initial belief state b_0 for all methods. We then expanded B iteratively using each of the heuristics described in Algorithms 6–9. The results of experiments with these heuristic are presented in Figures 4.5–4.10 for all the larger domains (hallway, hallway2, maze33, coffee, RockSample[4,4], and tag). The results reported here are based on several iterative expansions of B. In our experiments we also tried considering all beliefs which are only one-step away from the current set B. The number of backed up belief points is considerably larger in this case, so in principle this can lead to the most accurate approximation. However, the time to perform backups increased exponentially with the increase in $|B|$. In practice, we were not able to perform more than 2–3 expansions of B for the problem domains we studied and the solution quality did not improve much.

It is necessary to mention that in all the domains used, except the RockSample[4,4] and coffee, there is a high level of stochasticity [1]. In the RockSample[4,4] domain, actions are deterministic, and there is significantly less noise in the observations. The coffee domain has almost deterministic action transitions and slightly stochastic observations. The complexity of the optimal value function can be measured by the number of α-vectors to represent it. PBVI keeps at most one α-vector for each belief state in the set B since some of the belief points might share the same α-vector. The solution time of the algorithms can be measured in part by the size of B and the number of α-vectors representing the approximate value function.

Recall that in the design of the Algorithm 6 in Section 4.3, we defined a control parameter p. Conceptually, this parameter together with the discount factor γ determines how much the reachability of a point matters in our selection strategy. We have experimented

[1] A highly stochastic transition gives little information about the next state in transition from any state. In the extreme case of stochasticity a transition from any state gives a uniform distribution over all other states

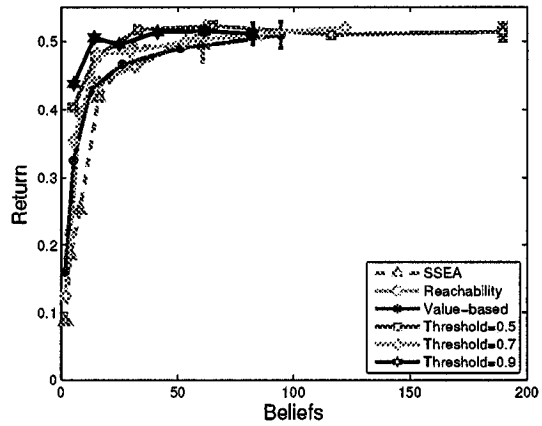
different values of $p \in [0,1)$, but we did not observe significant changes in the results ($\gamma = 0.95$ in almost all problem domains). Here we report the results with $p = 0.99$.

The threshold parameter e is a control parameter in the threshold-based heuristic Algorithm 9. We report experiments with $e = 0.5, 0.7$, and 0.9. In our implementation of the algorithm, we sample one observation for each action selected in the forward search from a point $b \in B$. In a highly stochastic domain of large size, such as the tag problem, with a small threshold e we generate too many samples, which can slow down the progression of the algorithm. Handling over 1000 belief points is very slow. Therefore, we have to trade off the accuracy of the approximation and the time, by moderating the distance control threshold e.

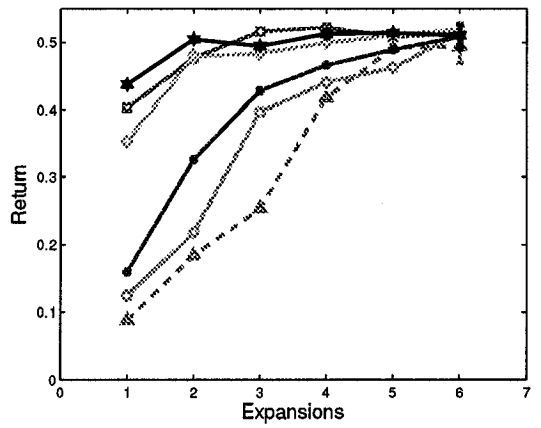
In the experiments, the set Γ_i in most of the domains contains as much as $|B|$ α, which confirms that the optimal policy has a huge size and perhaps we need to sample points densely in order to fit well this function. However, since the value function is piecewise linear and convex, the more facets it needs in its representation the smoother it becomes. However, with just a small subset of these facets we can reach a good approximation of such functions. Our results with CBVI in the previous section confirms the same fact. In the RockSample domain, only a small proportion of belief points in B offer distinct α-vectors. This is mainly due to the deterministic dynamics, which makes it possible to generalize the approximate optimal policy well enough over the whole space based on a small number of points. However, this does not necessarily suggest that with any small set of points we can reach a good solution. As we mentioned earlier, this suggests that the value function might have a sharp (or non smooth) shape which makes it difficult to find a proper subset of facets to represent it with good precision. A small size belief set chosen randomly might result in very poor performance. As we change the threshold e to higher values the difference between $|B|$ and $|\Gamma|$ is decreased confirming that we are considering more useful beliefs. We conjecture that a different way of facilitating exploration in the reachable belief space, perhaps by using our current estimate of the value function as a guideline combined with the distance feature, would help to attract more critical points into B.

The performance of all methods is presented in terms of the solution quality versus time since ideally we are interested in the approaches that make a good solution in a shorter time. In each figure we present the solution quality versus the size of the belief set B. As we mentioned before the size of B can be used as an indication of the time to get the ultimate solution, as well as an indication of the size of the ultimate value function. This is also used to compare the behavior of different methods in expanding their point sets with respect to the dynamics of problem domains. Recall that the PBVI algorithm is an anytime method to get an approximate solution. Obviously, the more expansions it does to the set B, the better solution it gives. However, sometimes it is important to know the solution quality in a particular cut off (after a particular number of expansions). Therefore, here we present graphs demonstrating the solution quality versus number of expansions in each domain as a subpart of each figure. In all graphs the error bars indicate the standard deviations.

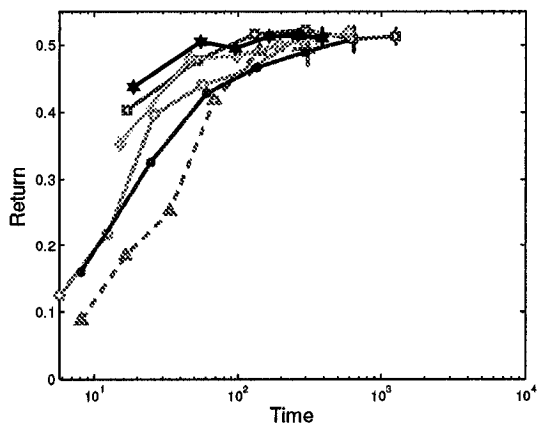
Figure 4.5 shows the performance of different methods on the hallway domain. The threshold-based method is the winner, with a significant difference in providing much better solutions in different expansions, such that with only two expansions of B with this method we can get to the convergence point. The difference between variety of values of the threshold e seems to be small. The performance of CBVI with only one-step core beliefs on this domain shows the importance of beliefs reachability and explains the reachability-based superiority to SSEA here. The value-based method also performs better than the SSEA. In our experiments we observed that the number of α-vectors and the number of belief states stay the same as we progress. This means that the optimal value function should consist of many facets which can only make its shape smooth. We speculate that the value-based approach can benefit from this smooth shape of the perspective optimal value function. The structure of the domain and the shape of the optimal value function makes the difference in the values of the candidate points more a sensible criterion than the difference in their geometrical distance. As we mentioned earlier in this chapter the maximum geometrical distance is not necessarily a good indication of the potential difference in the optimal value function.



(a)

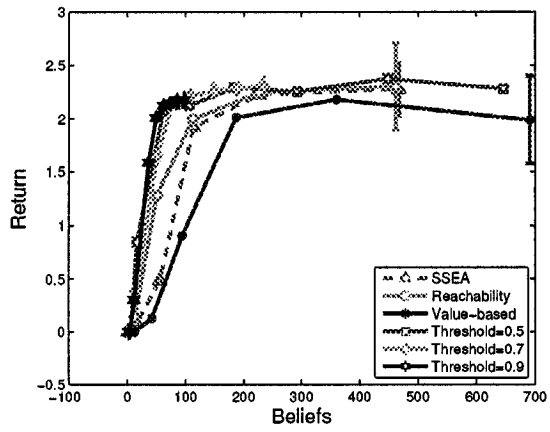


(b)

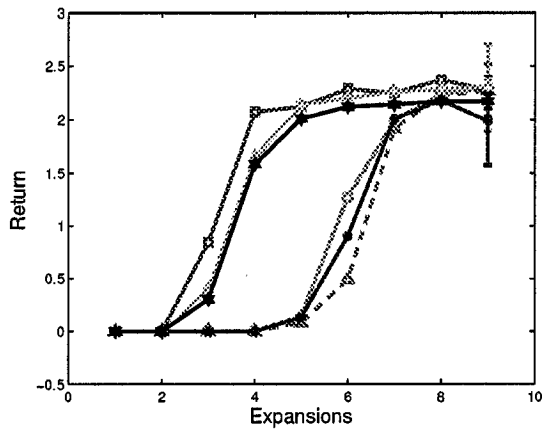


(c)

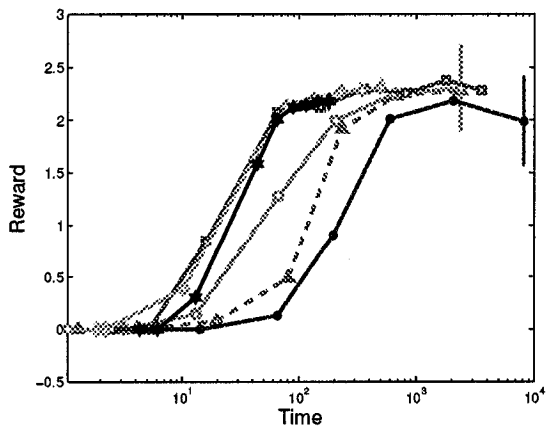
FIGURE 4.5. Policy performance for different belief point selection methods in the hallway domain.



(a)

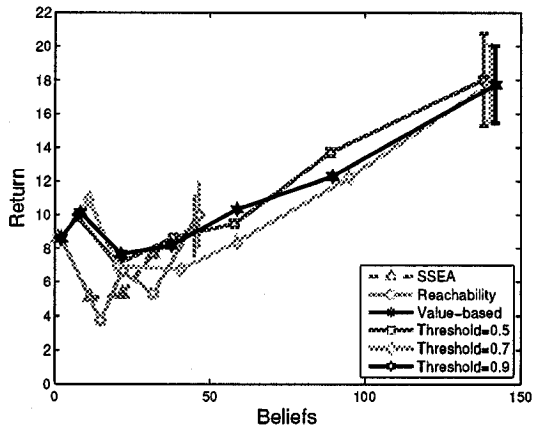


(b)

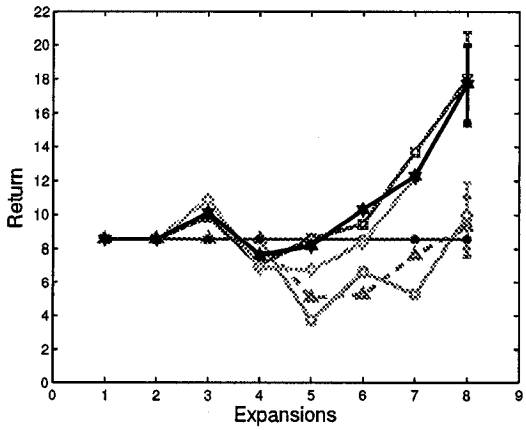


(c)

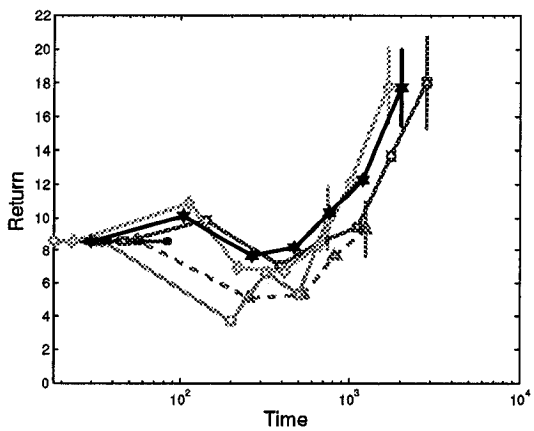
FIGURE 4.6. Policy performance for different belief point selection methods in the Maze33 domain.



(a)

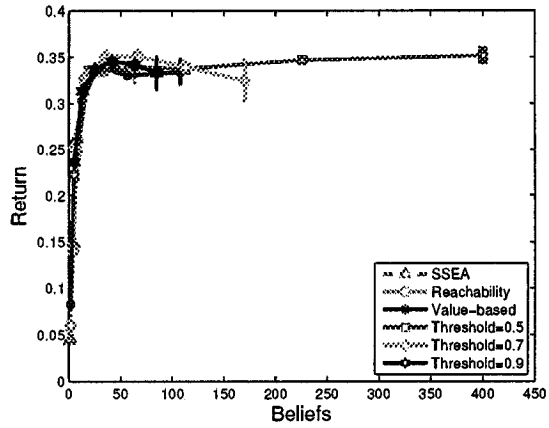


(b)

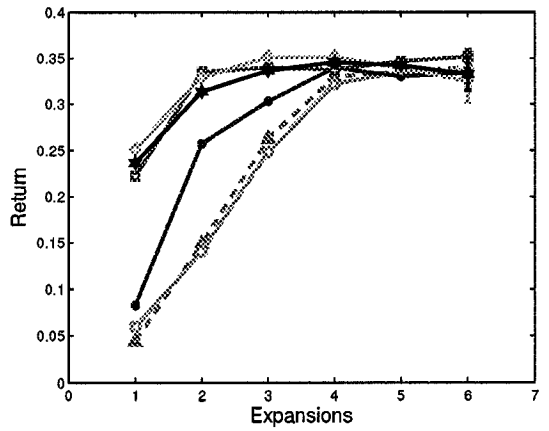


(c)

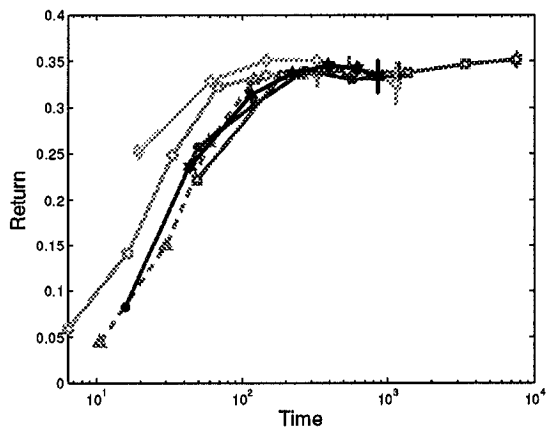
FIGURE 4.7. Policy performance for different belief point selection methods in the RockSample[4,4] domain.



(a)

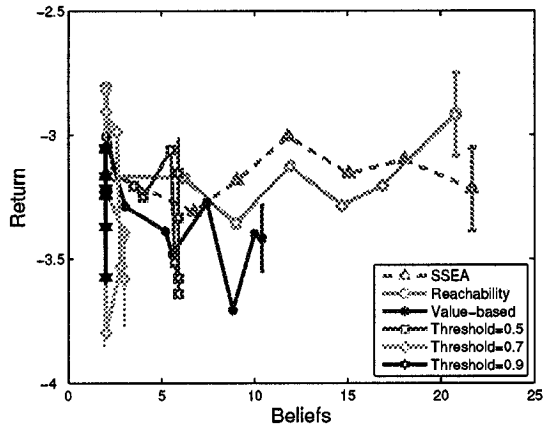


(b)

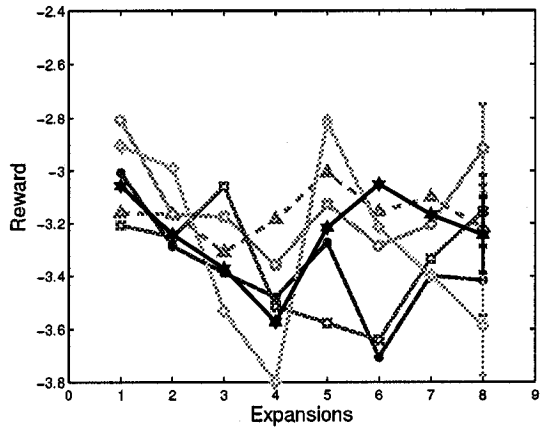


(c)

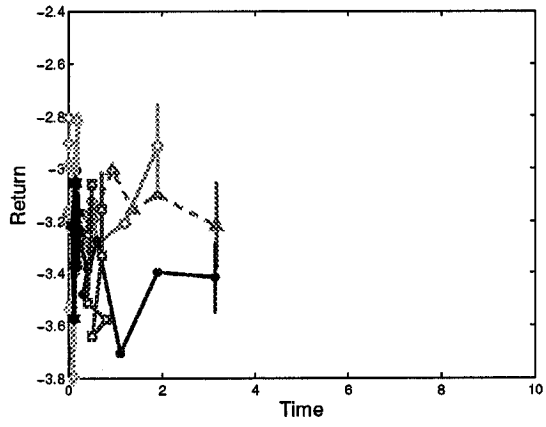
FIGURE 4.8. Policy performance for different belief point selection methods in the hallway2 domain.



(a)

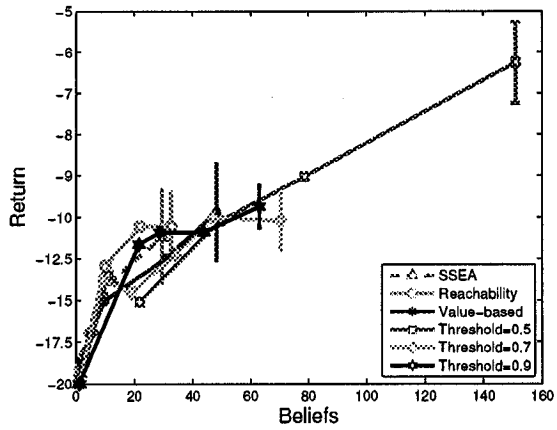


(b)

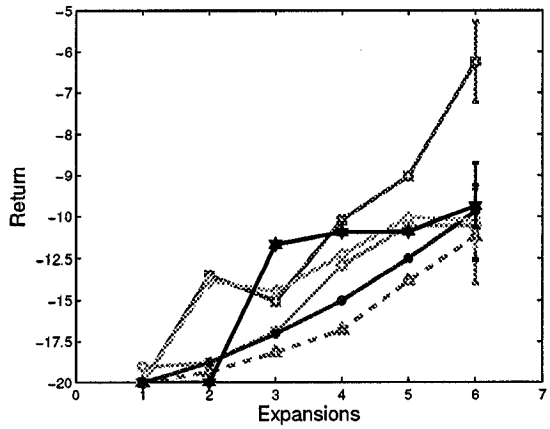


(c)

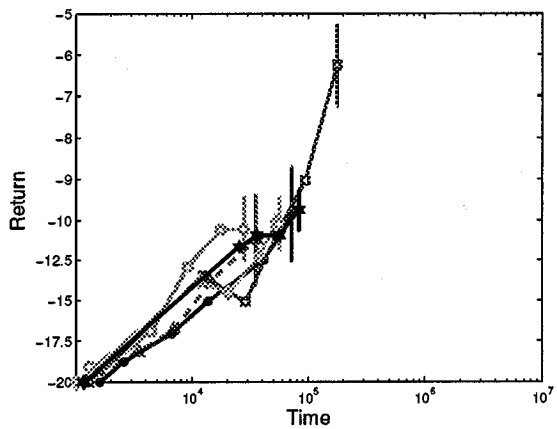
FIGURE 4.9. Policy performance for different belief point selection methods in the Coffee domain.



(a)



(b)



(c)

FIGURE 4.10. Policy performance for different belief point selection methods in the Tag domain.

The experimental results on the Maze domain is presented in Figure 4.6. In this figure we observe that the difference between reachability, value-based and the SSEA heuristics is not noticeable with respect to the expansions of B. However, some difference in the size of their belief set makes the reachability approach converges faster than SSEA, and the SSEA faster than the value-based. These differences are more pronounced some while after the beginning since except for the threshold-based method, all others perform very poorly at the beginning. The threshold-based approach performs considerably better than other approaches with all values of e.

For the case of RockSample domain, Figure 4.7 confirms again that the threshold-based method is superior in providing higher quality solutions. We are presenting 8 expansions of the set B in this figure. Only the threshold-based method can converge to the prospective optimal reward value. Higher number of expansions make other approaches converge to the same reward value. The value based method does not increase the size of its belief set and therefore stays in a locally optimal value function the algorithm progresses. There is always a considerable difference between the number of beliefs in B and the number of α-vectors representing the approximate value function using all the approaches. This can indicate that the optimal value function does not have too many facets and a smooth shape. This also confirms our reasoning for the performance of the value-based method.

Performance comparison of all methods is presented in Figure 4.8. This domain shares a similar structure with the hallway domain. There is a small difference in the threshold-based method with different e values in terms of the performance in each expansion. However, the threshold $e = 0.5$ seems to grow the size of B significantly, such that this technique can not produce any solutions earlier in time. There is a very small difference both in terms of the solution quality and the time required to reach these solutions between the SSEA and the value-based method. Reachability-based method outperforms other approaches for some while at the beginning. We realized that for the case of CBVI method in the previous section, we could reach a good approximation with core-beliefs reachable at the lower level in the forward search tree (1-step or 2-step length core-beliefs). This confirms that a

selection method such as the reachability-based provides a good set of points as we need to sample from points in earlier future to make improvements.

The performance of all algorithms are almost the same and strange in the case of the coffee domain in Figure 4.9. The value based method performs poorly in Coffee domain as well. The reachability selection criterion present small improvement for this case. As can be seen in the Figure 4.9(b) the size of the belief set does not change much as we change the threshold e. Although, the growth of B in other methods did not make much difference in the discounted reward either.

Figure 4.10 illustrate the experimental results on the tag domain. We have performed 6 PBVI belief set expansions for this domain. In this domain we can see that threshold-based heuristic, $e = 0.5$ becomes too slow after a while since in each expansions the number of belief points added to B grows too fast and considering the huge size of the state space this is very time consuming. On the other hand other methods do not seem to be efficient in improving the policy quality. The reachability-based in general outperforms other methods in the case of the tag domain. This could mean that beliefs reachable in earlier future are more important.

4.7.4. Discussion

According to the results presented in this chapter, in almost all domains we observe that all the methods converge to a common approximate solution at a different paces. The only exception is for the value-based heuristic. The value-based approach in some problems does not allow a good expansion of B and can be trapped in a local optima. RockSample domain is an instance of this form. This can be the effect of deterministic transitions in this domain which makes the choices for the belief point selection of this heuristic very limited. In general, the difference between most of the heuristics is difficult to narrow. Theoretical characterization of domains will be required to be able to decide which approach to choose. However, to this date there is no such results available in the literature. Although we can make speculations and reach some informal conclusions like the ones stated in the previous section. For instance, if the problem is very discounting (i.e. has a very small

discount factor) then perhaps reachability considerations is important when choosing between candidate belie points. Since the expansions of the set B potentially adds beliefs which are more far in future, small difference between the reachibility-based and the SSEA heuristics could mean that the uniform sampling of points in the reachable belief space and the average-norm sampling in this space do not matter much due to the structure of these domains.

In total, the effect of the threshold-based approach seems to be quite promising. This method obtain a higher quality solution with comparably smaller number of points. Although it should be noted that the choice of the right threshold is quite important to achieve the beneficial results. In our experiments, we tried several values of this threshold between the maximum and minimum norm-1 distance between candidate points $[0,2]$ randomly. Good choice of this threshold needs characterization of the underlying domains with respect to different parameters such as sparsity of the reachable belief space, the structure of the transition, observation and reward functions or the possible realization of the shape of the value function. Our results confirm that the exploration-exploitation trade-off in the space of reachable points impacts significantly the quality of the solutions obtained.

4.8. Summary and Conclusion

The class of point-based algorithms (PBVI and its variations) presented in this chapter introduces an efficient approach for scaling POMDP solution methods. In this chapter, we introduced and evaluated several point selection criteria for PBVI. First, we presented CBVI method for selecting a set of representative belief points which relies on the reachable basis for the belief simplex. CBVI theoretically provides a tighter worst case error bound than PBVI. We also demonstrated how this method can lead the approximate value function of PBVI to cover the belief space more quickly in domains with special structure. This algorithm consistently finds controllers which are competitive with other state of the art POMDP controllers.

The reachability criterion was examined in three levels: (1) high importance: very biased approach of selecting all points in breath (2) medium importance: selecting points in

breath but proportional to their $norm-1$ distance from B, and (3) no importance: considering only the $norm-1$ distance from B giving a uniform importance to the reachability distance. Our experiments showed that (1) is not usually a practical approach. It is difficult to distinguish a winner between 2 and 3, and it can only depend on the knowledge about the environment. We carried out similar study on different problems from the view point of geometric distance. This distance e was varied from the most conservatives approach, CBVI, for $e=2$, to the most relaxed approach, breath, for $e=0+\varepsilon$. However, in all cases performance was significantly better at the intermediate value of $0.5 \leq e \leq 0.9$. Thus we have a range of results which suggests that distance threshold metric is generally more efficient than the reachability distance metric. While it remains unclear exactly how to find the best value of e. We also tested the idea of using the value estimates as a guide to select points in a forward search. The exploration-exploitation trade-off seems to play an important role, and should be taken into consideration.

Dimensionality reduction techniques combined with point-based approximation methods are of potential interest in this context. We mentioned the approaches in the literature attempting to do so, however this is still a widely open research area. In Chapter 6 we will discuss the applicability of the point-based methods by predictive state representation and will target the both curse of dimensionality and history for the case of highly structured domains.

CHAPTER 5

State Space Structure and its Implication in Decision Making

Chapter Outline

Exploiting the special structure of problem domains is an important step in the design of a good controller. In this chapter we study this problem using the mathematical properties of predictive state representations. We analyze the connections between several POMDP compression techniques and predictive state representations. We study the utility of augmenting the state information with reward from the point of view of planning and state monitoring.

The intractability of planning in partially observable systems is due in part to the size of the domain. The number of variables needed to represent the state space, action space, and observation space has a great impact on the efficiency of planning. Although for many practical applications the number of variables describing these spaces is very large, these variables are not always distinct or not all distinctions are necessary. A typical means of overcoming such situations is to aggregate similar variables and make a smaller and more efficient model. The hope is that optimal values and optimal policies for the original model are similar to those obtained from the reduced model. On the other hand, we are interested in reducing the uncertainty by having a problem formulation as detailed as possible. These two goals involve a trade-off between the size of the model and the accuracy of the solution

that can be obtained. There exists a large body of literature on the trade-offs between primitive and abstract representations in learning and reasoning in general. In this chapter we discuss the issues relevant to decision making in partially observable domains.

Recall that many POMDP solution methods are based on computing and monitoring belief states based on a model of the environment, and the history of actions and observations seen by the agent. The action choice of the agent is then based on the belief state. If in the definition of the problem there exist states in which any behavior of the agent would result in the same outcome, or there are observations that are equally probable in all states under any actions, we would be interested to aggregate these variables to derive smaller equivalent models of the problem. In this chapter we argue that this aggregation should take rewards into account as well. Rewards can carry useful information, in addition to the observations, that can also help disambiguate the hidden state. The reward information should be considered in the POMDP state representation as an extra observation. The fact that rewards are only used in the computation of the optimal policy, but not in updating the belief state, is due to the fact that POMDPs have roots in MDPs. In an MDP, the agent is provided with Markovian state information directly, and uses the rewards in order to obtain an optimal policy. The same scenario has been carried out to POMDP planning methods as well, except now the Markovian belief state has to be recovered from the history, and the history includes the rewards received as well. In this chapter we study the effect of rewards on the belief state update.

The rest of this chapter is organized as follows. Section 1 presents notation, basic definitions, and reviews previous work on MDP compression. Section 2 focusses on the types of structure that can be exploited by predictive state representations in Section 3. We develop the connection between PSRs and other previous models for state aggregation. Section 4 discusses the issue of using reward as part of the information available for belief state updates, and provides an empirical analysis of this approach. Section 5 summarizes the chapter

5.1. Model Minimization in Markov Decision Processes

Model reduction in MDPs relies on state space compression (aggregating states); or composing actions (temporal compression). Spatial and temporal abstractions have been considered in MDP minimization framework by many researchers [Givan *et al.*, 2003; Giunchiglia and Walsh, 1992; Ravindran and Barto, 2002; Sutton *et al.*, 1999b]. State aggregation in MDPs is usually done by replacing one MDP by another smaller one which either ignores state distinctions that do not affect performance, or groups together states that behave similarly. State aggregation is considered for MDPs with respect to different criteria, for instance the transition functions, reward (or value) functions or both. The decision problem based on the abstract states, $x_1, x_2, ..., x_k$, is not Markovian in general. It can be viewed actually as a special case of POMDP, defined by a special set of observations $O = \{x_1, x_2, ..., x_k\}$, with the observation probability $P(x_i|a,s) = 1$ if s belongs to the abstract state x_i and 0 otherwise. Exact abstraction means that no information loss exists due to the abstraction process.

We focus only on types of abstraction in which the Markov property is preserved. *Bisimulation* [Givan *et al.*, 2003] is a very strict criterion for state aggregation. They generalize the definition of bisimulation from the process algebra literature [Larson and Skou, 1994] by incorporating reward equivalence as well. While this type of abstraction preserve the dynamics of the model in the abstracted version, there are other types of abstractions which are more relaxed and only preserve properties useful for decision making. For instance, [Dearden and Boutilier, 1997] create an aggregation which only considers preserving the optimal state-action value function in the abstract model.

A recent survey by [Li *et al.*, 2006] introduces a comprehensive taxonomy of abstraction methods on MDPs based on preserving five essential properties in MDP solution methods.

Definition: ([Li *et al.*, 2006]) Given an MDP $M =< S, A, T, R, \gamma >$, let the abstraction version of M be $M' =< S', A, T', R', \gamma >$, and let the abstraction function be $\phi : S \rightarrow S'$. For any states $s_1, s_2 \in S$, we define:

(i) A model-irrelevant abstraction ϕ_{model} is such that for any action a and any abstract state s': $\phi_{model}(s_1) = \phi_{model}(s_2)$ implies $R(s_1,a) = R(s_2,a)$ and $\sum_{s_3 \in \phi^{-1}_{model}(s')} T(s_1,a,s_3) = \sum_{s_3 \in \phi^{-1}_{model}(s')} T(s_2,a,s_3)$ where $\phi^{-1}_{model}(s')$ refer to the state s such that $\phi_{model}(s) = s'$.

(ii) A Q^{ϕ}-irrelevant abstraction ϕ_{Q^π} is such that for any policy π and any action a, $\phi_{Q^\pi}(s_1) = \phi_{Q^\pi}(s_2)$ implies $Q^\pi(s_1,a) = Q^\pi(s_2,a)$.

(iii) A Q^*-irrelevant abstraction ϕ_{Q^*} is such that for any action a: $\phi_{Q^*}(s_1) = \phi_{Q^*}(s_2)$ implies $Q^*(s_1,a) = Q^*(s_2,a)$.

(iv) An a^*-irrelevant abstraction ϕ_{a^*} is such that every abstract class has an action a^* that is optimal for all the states in that class, and $\phi_{a^*}(s_1) = \phi_{a^*}(s_2)$ implies that $Q^*(s_1,a^*) = \max_a Q^*(s_1,a) = \max_a Q^*(s_2,a) = Q^*(s_2,a^*)$.

(v) A π^*-irrelevant abstraction ϕ_{π^*} is such that every abstract class has an action a^* that is optimal for all the states in that class, that is $\phi_{\pi^*}(s_1) = \phi_{\pi^*}(s_2)$ implies that $Q^*(s_1,a^*) = \max_a Q^*(s_1,a)$ and $Q^*(s_2,a^*) = \max_a Q^*(s_2,a)$.

The state aggregation of ϕ_{model} is in terms of identical one-step transitions and rewards. The MDP model minimization of [Givan *et al.*, 2003] is of this type. ϕ_{Q^π} performs abstraction based on equal action-value functions for all policies. ϕ_{Q^*} groups together states that have the same optimal state-action value function; the representation introduced in [Dearden and Boutilier, 1997] can be considered of this form; the fourth category. ϕ_{a^*} include aggregation of states with identical optimal action and its value. McCallum's utile distinction method [McCallum, 1995b] falls into this class of abstraction. Finally, ϕ_{π^*} aggregates states with the same optimal action [Jong and Stone, 2005]. An interesting and important

[1] S and S' present the state spaces, T and T' the transition functions, and R and R' the reward functions of the original (ground) model and of the reduced model, respectively. The actions space is assumed to be unchanged in both models

observation drawn from this characterization is a partial order $\leq$ between abstractions defined in Theorem 2 of [Li *et al.*, 2006]. Abstractions are fined from left to right for any MDP, $\phi_{\pi^*} \leq \phi_{a^*} \leq \phi_{Q^*} \leq \phi_{Q^\pi} \leq \phi_{model}$. Therefore, the preserving properties of one abstraction automatically apply to its finer abstractions. For instance, ϕ_{model} inherits the preserving of action-value function for all policies from ϕ_{Q^π}. This is specially helpful when designing learning and planning algorithms. We refer to this categorization later on in Section 5.3.

Many approaches in the MDP literature are based on divide-and-conquer techniques in which the original MDP is divided into several smaller MDPs (abstract classes) [Wang and Mahadeavan, 1999; Singh, 1992b; Kaelbling, 1993; Dayan and Hinton, 1993]. Hierarchical methods are promising in solving large MDPs given that a hierarchy of subtasks or subgoals can be defined in the original MDP. The decomposition is usually performed by defining abstract actions (sometimes referred to as temporally extended actions, closed-loop policies, options or macro-actions) and the type of abstraction can be mainly thought of as ϕ_{Q^*} in the level of temporally extended actions. However, automated subgoal discovery in MDPs, can be a difficult task itself [McGovern *et al.*, 2001; Bakker and Schmidhuber, 2003; Mannor *et al.*, 2004; Simsek *et al.*, 2005]. The Max-Q algorithm [Dietterich, 2000b] is an example of MDP hierarchical abstraction which decomposes an MDP into several reduced MDPs with abstract states, abstract actions, and a new set of reward functions. The state abstraction of this method is coarser than ϕ_{Q^*}. PolCA [Pineau and Thrun, 2002] is a similar approach to Max-Q for hierarchical POMDPs, which also assumes a decomposition of subtasks is given beforehand. It must be noted that some of the MDP abstraction methods cannot be extended to POMDPs since they do not account for stochastic action effects.

Another group of minimization methods, known as ε-reduction techniques, slightly compromise optimality by aggregating states which have a small difference in transition probabilities and rewards under a given policy.

5.2. State Space Compression by Linear PSR

The properties of MDP abstraction methods discussed in the previous section relate to different types of structure which may be exhibited by the environment. Exploiting special structure requires algorithms and methods for discovering it. Not all abstractions can be created efficiently. In fact, many of these approaches require significant computational effort and include solving a hard problem of finding similar states just to extract useful information to use for abstraction. Discovering the abstract model efficiently is still an open problem for some types of MDP abstractions. Furthermore, the same kinds of structure do not directly extend to POMDPs, although the belief-MDP is Markovian. Some of these approaches can be extended to POMDPs to derive a reduced POMDP model by exhaustively searching for different kinds of structure. Here we show PSRs are able to find some kinds of structure in dynamical systems and discover the reduced model for any given input POMDPs. In fact the PSR-discovery algorithm in Chapter 3 gives the reduced model by PSRs as its output automatically. We derive a sufficient although not necessary condition that ensures compression by linear PSRs based on an interesting mathematical property of predictive state representations.

5.2.1. Linearly Dependent States

We introduce the notion of linear dependency in the underlying state space of an MDP with an example. Figure 5.1 illustrates a dynamical system with 5 underlying states 2 actions and 4 observations on the top part. The two graphs on the top present the two actions. One of the actions is deterministic (left) and the other one is stochastic (right) with 0.5 probability of reaching s_1 and 0.5 probability of reaching s_2 from states s_2, s_3, s_4, and s_5. Under any action, states s_3, s_4, and s_5 have identical transitions as s_2. Therefore the underlying state space can be compactly represented by the equivalent model on the bottom with only 2 states. This is a graphical view of the system representation by a linear PSR of dimension 2, i.e. with only 2 core tests (e.g. a_1z_1, and a_2z_2). This property, and consequently the related abstraction, cannot be discovered by usual methods for abstraction in the POMDP model.

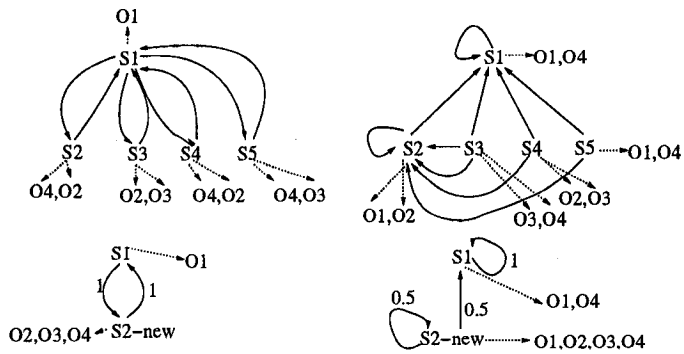


FIGURE 5.1. A POMDP problem with three linearly dependent states and four observation and two actions and its minimized model at the bottom

Definition: We say that a state s_i is *linearly dependent* on a subset $S' \subset S$ if $s_i \notin S'$ and its transition probability function under any action is a linear combination of the transition probabilities of states in S':

$$T^a(s) = \sum_{s_k \in S'} c_k T^a(s_k), \forall a \in \mathcal{A}, \tag{5.1}$$

where $0 \leq c_k \; \forall k$, and $\sum_k c_k = 1$.

THEOREM 9. *If the underlying MDP of a given POMDP has a linearly dependent state, then a linear PSR will provide a model with fewer core tests than the number of states in the original POMDP.*

PROOF. When deriving PSRs from the outcome matrix U as explained in 3.1.1, compression happens if and only if at least one of the rows of U is a linear combination of all others:

$$\exists i \text{ such that } U_i = \sum_{k \neq i} c_k U_k \tag{5.2}$$

therefore:

$$p(t_j|s_i) = \sum_{k \neq i} c_k p(t_j|s_k) \; \forall t_j \tag{5.3}$$

Suppose there exists a state $s_i \in S$ which is linearly dependent on a set of states S', so:

$$\forall a \in A : T^a(s_i) = \sum_{s_k \in S'} c_k T^a(s_k)$$

To prove that the ith row of U is a linear combination of the other rows for all possible tests, we proceed by induction. Consider first one-step tests. Let O^a be a matrix of size $|S| \times |O|$, giving the probabilities of different observations emitted from each state, after action a is taken. Then, by taking transposes and multiplying the above equation, we get:

$$(T^a(s))^T O^a = \sum_k c_k (T_k^a)^T O^a$$

Notice that this corresponds to the part of the ith row in the U matrix which contains the observations for all one-step tests for action a. Now suppose that we have established for all tests t of length l that the outcome of t in state i can be written as a linear combination of the outcomes of states k:

$$u_i(t) = \sum_{k \in S'} c_k u_k(t)$$

Consider a test aot of length $l+1$. We have:

$$u_i(aot) = T_i^a O^{ao} u(t) = (\sum_{k \in S'} c_k T_k^a) O^{ao} u(t) = \sum_{k \in S'} c_k (T_k^a O^{ao} u(t)) = \sum_{k \in S'} c_k u_k(aot)$$

Hence, the ith row of the U matrix is a linear combination of the rows corresponding to the states in S', with the same mixing coefficients as those from the transition matrix. Therefore the rank of U is strictly less than $|S|$. Since the dimension of the linear PSR is given by the rank of U, in this case the linear PSR representation will be smaller than the size of the state space. □

It must be noted that this theorem relates only linearly dependent state transitions to PSR compression without considering the observations. Notice the difference between the MDP abstraction methods in the previous section and PSR state abstraction. The type of state abstraction that the linear dependence property of PSRs provide maintains the dynamics of the system but not necessarily the values. In fact, compression in linear PSRs can happen due to other kinds of structure in the domains in particular related to a combination of the behavior of the transition and observation functions. For example, consider the problem of navigating in the environment shown in Figure 5.2. The 4-state MDP with deterministic transitions under taking two actions: turn clockwise or anticlockwise has no

linearly dependent state. However, the corresponding POMDP model, on the middle, with two deterministic observations can be minimized to a POMDP model on the right recovered by only two PSR core-tests. Considering the reward as a part of observation, the total

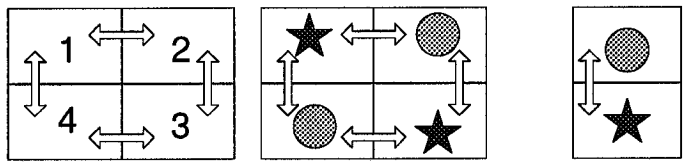


FIGURE 5.2. An MDP problem with 4 states and two actions for moving clockwise and anticlockwise represented in the middle by a POMDP with two observations. The reduced model in the right corresponds to the PSR representation of this system with two core tests

compression provided by PSRs will preserve the dynamics together with the value just as in state abstraction in MDPs. It is still not clear how to characterize the effect of observations which might lead to state abstraction by PSRs. A good example in the literature which illustrates this fact is the POMDP coffee domain first used in [Boutilier and Poole, 1996]. This problem has 32 states, 2 actions and 3 observations. This problem has only 4 linearly dependent states (i.e., the transition functions alone reduce the problem significantly). Further, the problem can be compactly represented by a linear PSR with only two tests if we do not consider rewards. But this is not a safe abstraction since it looses some of the reward information. It is necessary to mention that care must be taken when performing POMDP abstractions based on observations, in order not to lose valuable information regarding rewards. This is basically due to encoding only observations but not rewards in the belief state formulation. We discuss this problem in more detail in Section 5.4. In Chapter 6 we explain that PSRs should consider rewards as part of observation for control purposes. This provides a safe abstraction which preserves information about values. In this case the coffee domain is represented by 11 core tests.

The linear dependence property is not an equivalence relation. Therefore, it does not generally partition the state space as in MDP state aggregation.

5.2.2. Special Case: Matching Transitions

The type of state aggregation that a linear PSR model can perform with respect to the underlying states s and s' in a POMDP, is that performing any action a in state s looks the same (i.e. results in the same observations) as performing that exact action from state s'. This does not mean that the destination states are exactly identical positions, only that these states cannot be distinguished by any possible tests. It is interesting to know the type of structure which can be captured by PSRs in the special case of linear dependency that the transitions from some states are completely matched under taking any action. We use some examples to describe this type of structure. Consider a 4-dimensional grid world in Figure 5.3 which has identical states with respect to the observations (labels). The related PSR can capture this structure with the defined actions: *turn clockwise; turn counterclockwise.*

In general, many states can be aliased in terms of the immediate observations (i.e. can not be distinguished by one-step tests). However, as we consider experiencing longer tests, more states may become distinguishable by producing distinct sequences of observations. Although the special case of linear dependency looks more limiting, it is more likely in practice. Therefore, to capture it in the state representation can be advantageous specially in larger domains. For instance, a navigation problem in the environment shown in Figure 5.4 has only two observations *black* and *white* and two actions *turn left; turn right.* This environment has 576 underlying states (states are position-orientation pairs). Therefore, if we had a chance to take only one-step experiences with the environment we would recognize only two states, but if we consider infinite-length experiences there are 288 recognizable states shown in the lower grid in this figure. The two arrows shown in the upper grid belong to one class of states that produce equal observation sequences given any sequence of actions. These states are equivalent. The linear dependence property in the special case defines this equivalence relation. If the environment exhibit this type of behavior then the state space can be partitioned to equivalence classes. Of course the compression with this type of structure exploration in best cases is at most linear with linear PSRs. However, EPSRs which focus only on the last distinguishing observation, can be more efficient in defining less number of equivalent classes and therefore more compression. [Rafols *et al.*,

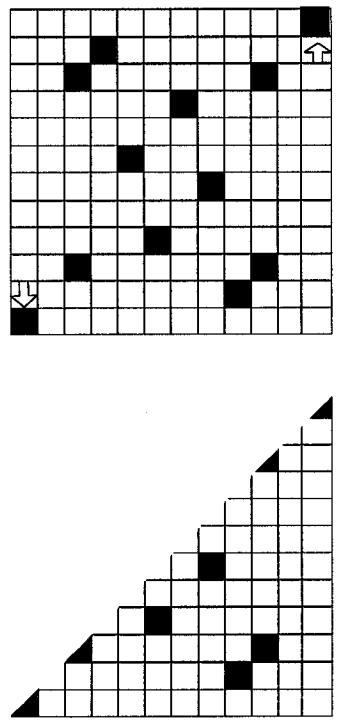


FIGURE 5.3. A POMDP problem with two observations: *black* and *white*; two actions:*turn left* and *turn right*; and 576 states (state=(grid position,orientation)) at the top. Arrows present a pair of undistinguishable states under all possible tests. The abstract MDP model is shown at the lower figure.

2005] show this type of partition for the case of EPSRs. It is interesting to note that the way the actions are formulated is important in forming the symmetric structure recognized by PSRs. For example in Figure 5.3 and 5.4 if we encode the actions for navigating in these grids as: *North, East, South, West*, then the resulting PSR model fails to discover the symmetric structure in the underlying state space of these environments. This is the difference in the definition of symmetric structure in PSRs compare to state-based representations like MDPs. It is realized that we need to encode a symmetric structure in the action space as well in order to discover the symmetry in the underlying state space. For instance, the action *turn right* is reversible by the action *turn left*, whereas action *North* can not be reversed by the actions *East* or *West*. Nonetheless, none of these symmetries can directly be noticed by the POMDP model before finding the minimized underlying MDP.

5.3. Related Work

In this section we describe other techniques for POMDP state abstraction that provide linear dimensionality reduction. Although these methods are all based on linear algebraic operations on state transitions and/or reward functions they focus on preservation of different properties.

5.3.1. Value-Directed Compression

The value-directed compression (VDC) algorithm [Poupart and Boutilier, 2003b], computes a low dimensional representation of a POMDP directly from the model parameters: R, T, O, by finding a *Krylov* subspace for the reward function under propagating beliefs. The Krylov subspace for a vector and a matrix is the smallest subspace that contains the vector and is closed under multiplication by the matrix. The smallest subspace that contains the immediate reward vector and is closed under a set of linear functions defined by the dynamics of the POMDP model generates the compressed belief space. The value-directed compression method has differences as well as similarities to predictive state representations. Both PSRs and VDC provide linear compression and they are both considered as lossless compression methods. Their approach for recursively growing a set of sufficient variables for prediction is identical. However, they focus on preserving slightly different properties in their compression as we explain below.

VDC exploits three types of structures namely *conditional independence*, *context-specific independence*, and *additive separability* in POMDPs viewed as a Bayesian Networks. In general, conditional independence allows discovering regularities in the conditional probability tables for variables in a Bayesian Network. Context-specific independence defines irrelevance of some variables in a specific context and additive separability specifies subsets of variables such that the marginals of each subset are sufficient to predict the marginals of the same subsets at the next time step. In fact VDC considers the decomposition of the value function into independent reward components. In VDC the belief state b is compressed to $\hat{b}$ and then after a transition the next belief state b' might not be correctly recovered since features of this state irrelevant to computing the value might be lost during

the compression. In PSRs, the space of beliefs spanned by core beliefs corresponding to the predictive state space can be viewed as a linear lossless compression of the original belief space. PSR abstraction ensures the accurate prediction of all future observations, whereas VDC abstraction ensures the accurate prediction of future rewards. Therefore, a next belief state in the original POMDP can be correctly recovered by the PSR. If reward is considered as part of observation, then PSRs focus on the accurate prediction of observations and rewards. Not including rewards as observations can lead to loosing information about values in the PSR representation. Note that in most POMDP problems, observations and rewards carry the same information about the underlying hidden states. However, this is not a requirement in the POMDP framework. Observations and rewards can provide different information about the states, independently.

The type of transformation matrix for VDC: $F = \{R, TR, T^2R, ...\}$ can be thought of as a change of basis for the value function, whereas the transformation matrix $F = U^T$ for the PSR model can be viewed as change of basis for the belief space. This implies that VDC does not lose information with respect to the value function and PSRs do not lose information with respect to the dynamics. If rewards are included as observations, PSRs do not lose value function information either, but may consequently provide less compression.

5.3.2. Trace Equivalence and Linearly Dependent States

The bisimulation relation has very strong assumptions which often do not allow for a lot of compression. This is a motivation to define more relaxed extensions of this relation. *Trace equivalence* between two process is defined in verification literature [Jou and Smolka, 1990]. Two processes are trace equivalent if and only if they yield the same probability distribution on observation sequences. Trace equivalence is less distinguishing than bisimilarity: two bisimilar states are guaranteed to be trace equivalent, but not vice versa. In the context of POMDPs, two underlying states s_1 and s_2 are trace equivalent if and only if under any sequence of actions they generate the same probability distribution on observations. This exactly complies with the required condition to have state aggregation imposed by PSRs. Therefore, trace equivalence can capture the same type of structure as linear PSR.

While exact equivalence is still limiting, Desharnais et al. [2006] have recently designed an approximate equivalence utilizing a metric based on the Kullback-Leibler divergence between the probability distribution on observation traces generated from two states upon taking the same sequences of actions. Treating rewards as part of observations, this metric preserves approximate values. Ferns et al. [2005] also suggested several metrics for measuring approximate bisimulation in MDPs which respect the transition as well as reward equivalence. It would be interesting to develop similar metrics for approximate PSRs which do not distinguish between tests that are close in terms of their probability after any history.

5.4. Belief State and the Effect of Reward Function

It is possible that for some problems, rewards are informative in state estimation. However, even when this is true, rewards in POMDPs are still used only in the policy computation, and not encoded in the belief state. Intuitively, it seems that rewards can carry useful information, which can help the agent guess the hidden state more precisely. This could well be the case when there is a distinction between the collection of rewards possible for taking an action in different states and the collection of observations made through this transition.

Indeed, in predictive state representations rewards can be used as part of the observation vector to update the PSR state of the agent. We present a method for updating the belief state which takes rewards into account.

5.4.1. Reward-Based Beliefs and RPOMDP

Intuitively, if the hidden state were known with better precision, the action choices of the agent could be better as well. The convexity of the value function makes the expected reward lower towards the center of the belief space. Therefore, the higher the entropy, the closer to the middle of the belief simplex the state is and the lower the expected reward. By modeling the reward in the state representation we allow the planner to access more

information and explore more thoroughly areas closer to the edges of the belief space. Several approaches have been designed in the POMDP literature which suggest belief entropy reduction as a means for more efficient planning. For instance, heuristics proposed in [Cassandra *et al.*, 1996], and [Roy, 2003], address the need to model information gathering actions and refine the belief state.

As described in the previous chapters, rewards in POMDPs are given as a function $R(s,a,s')$, which depends on the current state, the next state and the action taken. Hence, we can treat rewards as random variables, with a conditional probability distribution $P(r|s,a,s')$, where $P(r|s,a,s') = 1$ if and only if $r = R(s,a,s')$, and 0 otherwise. Note that if we had additional information about the distribution of immediate rewards, instead of just knowing the expected values, this could be naturally incorporated in this framework.

If there is only a discrete, finite set of reward values possible in the MDP, $\{r_1, r_2, ...r_k\}$, where each r_i represents the immediate reward for taking some action a from some hidden state s, this probability distribution can be easily specified using a table. We note that in most POMDP examples, the number of possible immediate rewards satisfies this assumption, and is often very small. However, if this assumption is not satisfied, e.g. if rewards are continuous, a conditional probability distribution over rewards can still be specified in some parametric form, based on the given model of the POMDP.

We note that rewards and observations are conditionally independent given the current state s, the next state s' and the action a. From now on we will treat rewards in the same way as observations in predictions about the future. The definition of a history will be extended to include the rewards: $h = a_1 z_1 r_1 ....a_n z_n r_n$.

We define belief state updates based on rewards and observations, as follows:

$$\begin{aligned} b'(s') &= P(s'|r,z,a,b) = \frac{P(b,a,s',r,z)}{P(b,a,r,z)} \\ &= \frac{\sum_{s\in S} b(s)T(s,a,s')O(a,s',z)P(r|s,a,s')}{\sum_{s'\in S}\sum_{s\in S} b(s)T(s,a,s')O(a,s',z)P(r|s,a,s')} \end{aligned}$$

Value functions are computed in this case analogously to the case of regular beliefs. We call this model Reward-based POMDP (RPOMDP). As we mentioned previously, rewards

have been used by Poupart and Boutilier [2003b] in designing a compression method for POMDPs. The RPOMDP model is close to the value-directed POMDP compression algorithm by Poupart and Boutilier. This reward-based definition of belief together with the concept of core beliefs can be considered as a compression method for POMDPs which preserve the value and the dynamics of the original POMDP model as discussed in the previous section. Considering the core beliefs without reward information can damage the value function in the sense that the compressed model has a completely different value function than the original POMDP.

In the following section we present experiments with exact and approximate planning methods on several standard POMDP domains, using this belief update method, and show that it can provide advantages, both in terms of speed and in terms of the quality of the solution obtained.

5.4.2. Empirical Evaluation of RPOMDPs

In this section we focus on studying the effect of the RPOMDP model on two issues. We selected 5 domains from the POMDP literature. Table 5.1 lists these problems with their descriptions. The last column in the table shows the number of discrete reward values defined in the domains. We chose three domains in which adding rewards is more informative than using just observations. These domains are: Network, line4-2goals and coffee. The other two domains are either ones in which rewards provide the same information as observation (4x4grid) or rewards have more information than observations only for special actions (shuttle). First, we measure belief state entropy, to show that including rewards in the belief updates can reduce the uncertainty of belief states. Second, we analyze the quality of the solutions obtained using the RPOMDP formulation.

Impact of RPOMDP Model on Belief State Entropy

The entropy of the belief state is defined as:

$$H(b) = -\sum_{i=1}^{|S|} b(i) log_2(b(i)) \tag{5.4}$$

TABLE 5.1. Domains used in the experiments

Domain	$\|\mathcal{S}\|$	$\|\mathcal{A}\|$	$\|\mathcal{O}\|$	$\|\mathcal{R}\|$
line4-2goals	4	2	1	2
network	7	4	2	6
4x4 grid	16	4	2	2
shuttle	8	3	5	3
coffee	32	2	3	12

Note that this measure is correlated with the distance between the belief state and the edges of the belief simplex. We performed a set of experiments to test the effect of the RPOMDP model in reducing the uncertainty about the states on selected domains. The entropy of the agent's beliefs has been measured for both the POMDP and the RPOMDP model during 100 time steps running the same random policy. Figures 5.4 to 5.8 show the result of this experiment. The presented graphs are averages taken over 5 independent runs. For the network, line4-2goals and coffee domains, the uncertainty of RPOMDP beliefs decreases considerably and it stays always lower than the entropy of POMDP beliefs. Shuttle and 4x4 grid do not show a noticeable difference for POMDP and RPOMDP running a random policy to explore in reachable belief space. This is expected since rewards carry little or no information in addition to observations for some of the actions.

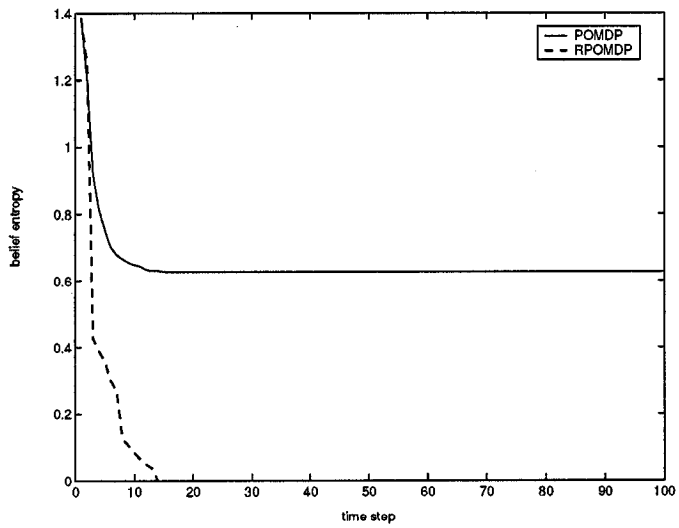


FIGURE 5.4. Line4-2goals profile. Comparison of the uncertainty in the agent's state of the world, using standard POMDP belief update vs. RPOMDP belief update.

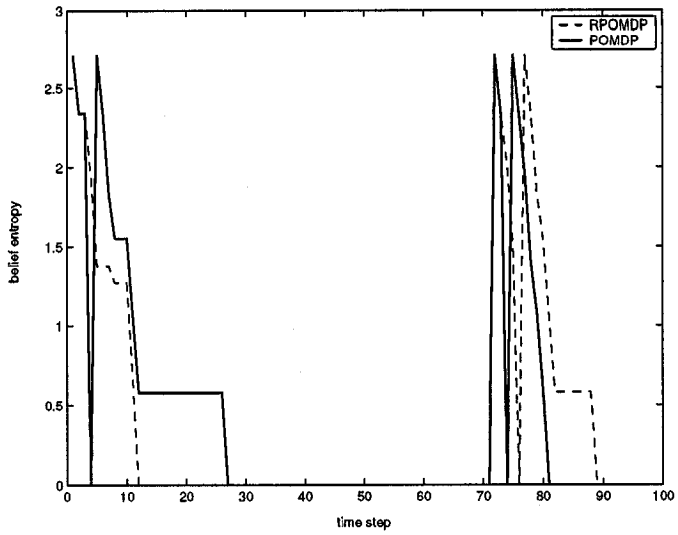


FIGURE 5.5. 4x4grid profile. Comparison of the uncertainty in the agent's state of the world, using standard POMDP belief update vs. RPOMDP belief update.

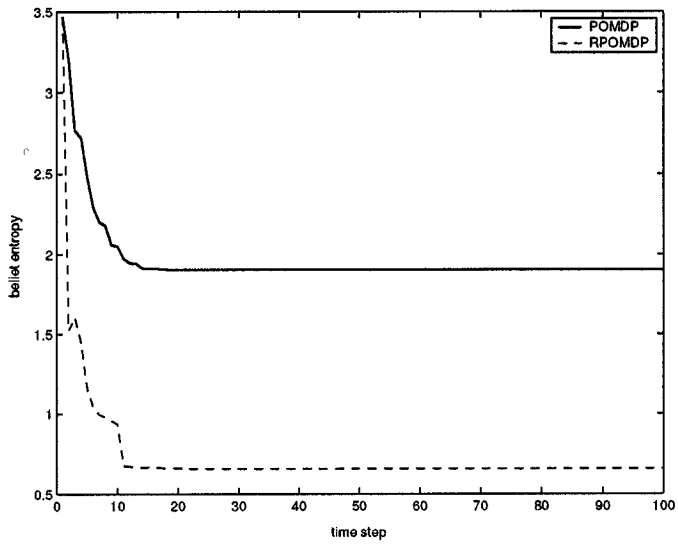


FIGURE 5.6. Coffee profile. Comparison of the uncertainty in the agent's state of the world, using standard POMDP belief update vs. RPOMDP belief update.

Impact of RPOMDP Model on Planning Methods

In this part of the experiments we used exact and approximate POMDP solution techniques to evaluate the performance of the RPOMDP model in terms of time to reach an optimal solution, the reached optimal value functions, and the complexity of the optimal value function with respect to the number of α-vectors, and to represent it.

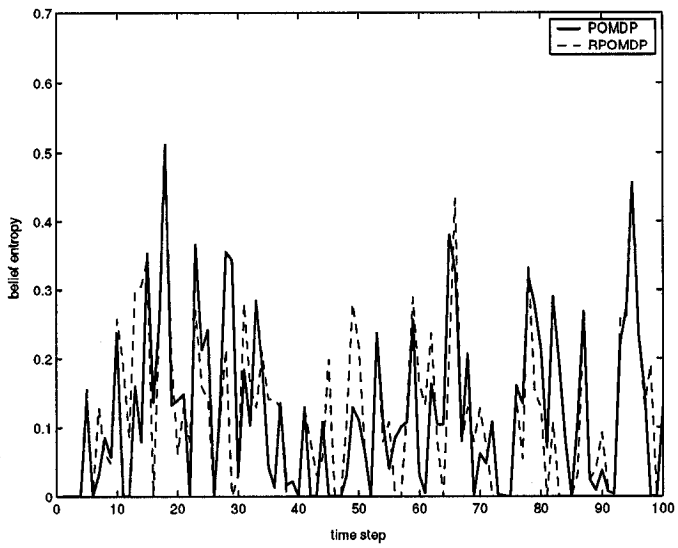


FIGURE 5.7. Shuttle profile. Comparison of the uncertainty in the agent's state of the world, using standard POMDP belief update vs. RPOMDP belief update.

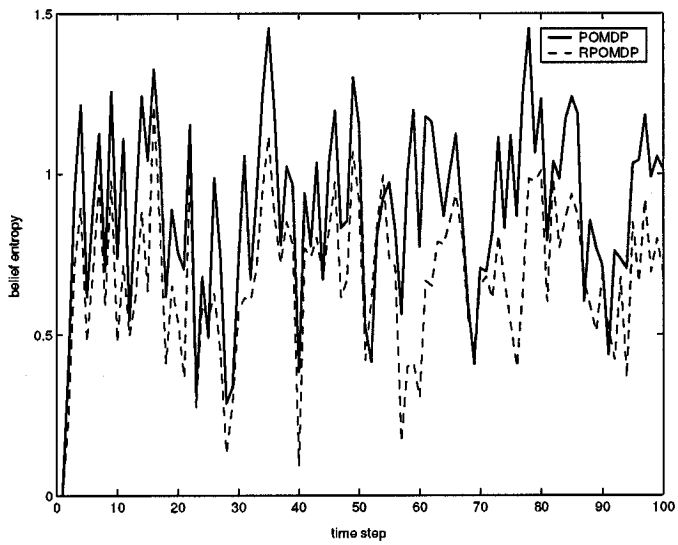


FIGURE 5.8. Network profile. Comparison of the uncertainty in the agent's state of the world, using standard POMDP belief update vs. RPOMDP belief update.

To find the optimal solution we used the witness algorithm described in Chapter 2 as an exact solution method for POMDPs. We ran the witness algorithm on all of the domains, but this algorithm is still incapable of finding an optimal solution in a reasonable time (5 hours of running time) for some problems such as shuttle and coffee. Table 5.2 contains the results of this part of experiments for comparison between the POMDP and RPOMDP

model using witness both for finite horizon $h = 10$ and for infinite horizon (large horizon length which makes the discounting reward $\gamma^h * r$ less than a pre-defined $\varepsilon = 0.0001$). This table reports the time taken to get the optimal solutions for both models; the size of the optimal value function as the number of α-vectors presenting it, and the value of an initial uniform belief state. Table 5.2 shows no difference in the case of line4-2goals problem for the value function representation and value received, however the time to get the optimal solution is greater in the POMDP than in the RPOMDP. This is expected since the state uncertainty for this problem disappears after a few time steps of even a random policy as shown in Figure 5.4. for the network problem, the observation space (containing rewards) becomes much larger in RPOMDP which explains the time difference in Table 5.2. The value functions for the two models are also different since the belief spaces are different. For the coffee domain, the witness algorithm did not proceed more than 14 epochs of value iteration, in the time allowed, while it can solve the RPOMDP very quickly as can be seen from this table. It should be noted that the true beliefs in this case occur in a very low dimensional manifold due to exhibiting lots of structure in the dynamics of this problem and it seems that RPOMDP can take advantage of this problem structure in the RPOMDP model of 4x4grid the space of observations (containing rewards) is larger than that of the POMDP model. However, observations and rewards carry the same information about the underlying states, so the belief spaces of the two models are the same which means that the value functions stay the same. However, there is a little bit of overhead in computations for the RPOMDP version. The witness algorithm did not terminate for the shuttle problem in both POMDP and RPOMDP in the given five hours time. However, from the results for the finite horizon we can observe that RPOMDP converges faster to the optimal solution. Of course the optimal value function of the two models must be different because of having different belief spaces.

We also used PBVI to evaluate the performance of RPOMDP in approximation methods.

The results of this evaluation are shown in Table 5.3. In all cases, we performed 10 belief set expansions of PBVI to obtain a value function. The time reported in this table is

TABLE 5.2. Performance comparison of exact solutions for POMDP original model and RPOMDP model

Domain	infinite horizon POMDP	infinite horizon RPOMDP	finite horizon POMDP	finite horizon RPOMDP
line4-2goals				
time	0.01	0.01	0.00	0.00
α-vectors	2	2	2	2
reward	0.466	0.466	0.465	0.465
network				
time	4539.15	5783.02	2.89	3.24
α-vectors	487	549	197	216
reward	293.185	340.15	121.27	173.94
coffee				
time	-	6.35	103	0.23
α-vectors	N/A	5	263	5
reward	N/A	0.00	0.00	0.00
4x4 grid				
time	5578.5	6459.3	231	316
α-vectors	243	243	245	248
reward	1.209	1.209	1.073	1.073
shuttle				
time	-	-	1270	81.4
α-vectors	N/A	N/A	2011	1152
reward	N/A	N/A	11.974	11.237

the time taken by the standard PBVI algorithm (SSEA) to reach an approximate solution. We also report the number of α-vectors for each model. The table presents the discounted total reward of trajectories of 250 time steps obtaining by running the approximate policy from the PBVI algorithm. These results are averages over 250 trials in each run and over 5 independent runs. The size of the belief set, B, is also shown in this table. The solution time for PBVI is directly related to the size of B and the number of α-vectors. Slight differences in time can be ignored since the experiments have been performed on different machines. The solution time increases for the RPOMDP model in the network problem for the same reason as in the case of exact solution. In the other problems the size of the belief set and the number of α-vectors are smaller or almost equal for both POMDP and RPOMDP. The solution quality in the coffee domains is different for the two models due to the decrease of the uncertainty about the states. The shuttle and the 4x4grid problems reached almost the

TABLE 5.3. Performance comparison of approximate solutions for POMDP original model and RPOMDP model

Domain	POMDP	RPOMDP
line4-2goals		
time	0.00	0.00
α-vectors	2	2
beliefs	232	157
discounted reward	1.24	0.48
network		
time	2	62
α-vectors	23	208
beliefs	714	500
discounted reward	240.63	352.9
coffee		
time	4	6
α-vectors	24	6
beliefs	323	169
discounted reward	-1.92	-0.97
shuttle		
time	0.8	0.01
α-vectors	17	18
beliefs	122	125
discounted reward	32.95	32.96
4x4 grid		
time	2.4	8.2
α-vectors	24	24
beliefs	460	468
discounted reward	3.73	3.75

same solution quality in both models as expected. The experimental results in Table 5.3 confirm that in the cases where rewards gives more information about the states than just the observations, significant improvements are achieved with respect to time. For the cases in which rewards do not matter in reducing the belief state entropy, there is no need to get them involved in belief updates. The RPOMDP in these cases might not change the belief space but it can increase the computation time.

5.5. Summary and Conclusion

POMDP solution methods deal with vectors of the size of the state space. State aggregation is a key idea for decreasing the complexity of these methods. In this chapter we defined a notion of linear state dependency between the underlying MDP states. We proved that linear state dependency is a sufficient condition for the linear PSR to have smaller dimensionality. However, if the underlying MDP is minimized in this sense and also for all actions a and observations o the matrix $T^a O^{ao}$ is non-singular, then it can be shown that the linear PSR space will be of the same dimension as the corresponding POMDP space (i.e. $|Q| = |S|$). It should be noted that the reverse is not necessarily true.

Our preliminary study illustrates this type of structure in the state space via transition functions which cannot be exploited by POMDPs but are detected by PSRs. By taking advantage of this kind of structure, linear PSRs reduce the effective size of the state space and can compactly represent certain dynamical systems. This simple fact opens up new possibilities for other compression analysis. We have not studied the effect of the reduced model on the value function.

While the compressed model is still a perfect model that can answer any questions about the system and make any predictions about the future of the system, linear compression is still considered inefficient in practice. This is a motivation to further investigate predictive models that can answer only task-specific questions, but perhaps scale better. Recent studies on EPSRs [Rudary and Singh, 2004] and PSRs with options [Wolfe and Singh, 2006] are early steps in this direction.

We also studied a reformulation of the POMDP model, focusing on the assumption that rewards can carry information about the states of the world independently of the observations. Following this assumption we represent and update belief states taking into account the reward signal as part of the feedback that the agent receives from the environment in order to reduce the uncertainty about the state of the world. We presented the results of an empirical study confirming that the RPOMDP model is very useful in reducing the entropy of beliefs for some domains. This can produce a significant performance improvement in solving POMDPs for those domains. Belief state entropy can be used in

order to guide more intelligent exploration strategies and reason about which parts of the continuous belief space are worth exploring. Often the evolution of beliefs following a trajectory is embedded in a low dimensional space. In the case of RPOMDPs, applying linear dimensionality reduction techniques involves no information loss. It would be interesting to study further nonlinear compression techniques using RPOMDPs.

CHAPTER 6

Planning with Predictive State Representations

Chapter Outline

In this chapter we address the problem of planning with predictive state representations. We present two methods for planning with linear PSRs. The first method is an exact planning algorithm based on ideas similar to classical look-ahead search. The second algorithm is an approximate planning method, which provides an extension of point-based value iteration (PBVI) to PSRs. This approach turns out to be a natural match for PSRs. We provide empirical results which show that these methods are either comparable or better than POMDP planning algorithms.

Decision making and control of dynamical systems involve taking optimal actions based on the current state of the system. The state must be a sufficient statistic for the history of the system that allows accurately predicting its future. Recall that the state of a system in PSRs is described by the predictions for a set of core tests given histories, $P(Q|h)$. The control problem for PSRs can be defined as follows: given the current prediction for core tests, decide what action should be taken in order to maximize the total return in future time steps. The original PSR design as discussed in Chapter 3, did not include a notion of reward. Therefore, reward must be defined explicitly in this representation in order to use it in decision making problems.

This chapter focusses on the PSR control problem. The main contribution of this chapter is presenting the first PSR control algorithm, published in [Izadi and Precup, 2003], as

well as designing a point-based approximate planning with PSRs. We describe the proposed algorithms in Sections 6.1, 6.2, and 6.3. Empirical evaluations of these methods are presented in Section 6.4. We discuss potential advantages of using this representation in control problems and relate our methods to existing approaches for exact and approximate planning under uncertainty. Conclusions and directions for future work are presented in Section 6.5.

6.1. Modelling Rewards in PSRs

The standard PSR framework does not consider rewards at all. However, these can be incorporated naturally by considering rewards as an extra component of the observations.

Formally, given a finite set of real-valued rewards $R = \{r_1, r_2, ...r_m\}$, we can define the reward received upon taking an action a given a PSR state as ρ:

$$R(\rho, a) = \sum_r rP(r|\rho, a) \tag{6.1}$$

where ρ is given by a prediction vector for the core tests in Q given the current history h: $\rho = P(Q|h)$.

Recall that the PSR model parameters consist of the core tests Q, vectors m_{az} for predictions of one-step tests, and matrices M_{az} for predictions of one-step extensions to the core tests. Define a binary ($|S| \times |S|$) matrix R^{ar} for each action-reward pair ar, such that each entry $R^{ar}(i,j)$ of this matrix is $P(r|s_i, a, s_j)$, the probability of receiving reward r upon taking action a from state s_i and landing in state s_j. This probability is equal to 1 when there exists a reward r for transition from state s_i under action a to state s_j in the given POMDP, and is zero otherwise. Rewards are defined in the POMDP model as depending on state-action pairs, but this dependency is interpretable in terms of action-observation pairs (tests) as follows. The scalar value r is observed with a probability of transition to state s_j, given the action and the departing state s_i, regardless of the observation received. However, the observation is generated based on the destination state s_j and the action a.

Therefore:

$$\begin{aligned} P(r,z|s_i,a) &= \sum_{s_j \in S} P(s_j|s_i,a)P(r,z|s_i,a,s_j) \\ &= \sum_{s_j \in S} P(r|s_i,a,s_j)P(s_j|s_i,a)P(z|s_j,a) \end{aligned}$$

This is a reformulation of the outcome functions defined in 3.1.1 to compute core-tests which contain rewards as: $q = a_1(r_1z_1)...a_n(r_nz_n)$. Consequently, the projection vectors and projection matrices of the form m_{az} and M_{az} will become of the form m_{azr} and M_{azr} respectively.

Now we can consider a scalar reward for every prediction vector ρ, upon taking an action a, and rewrite the Equation (6.1) as:

$$R(\rho,a) = \sum_r r \sum_z \rho^T m_{azr} = \rho^T (\sum_r \sum_z r m_{azr}) \tag{6.2}$$

For ease of notation let:

$$\eta_a = \rho^T (\sum_r \sum_z r m_{azr}) \tag{6.3}$$

Therefore:

$$R(\rho,a) = \rho^T \eta_a \tag{6.4}$$

This is similar to the definition of reward of taking action a at a belief state b in a POMDP.

6.2. Planning Formulation in Predictive State Space

A policy component of a PSR agent must map predictive states to an action given a sequence of action-observation-reward that took place prior to the current time (history). An agent can predict the effects of executing action a for the prediction vector ρ_t at time t as follows. First, we define a new notation for the observable feedback of the form zr, to include both rewards and observations, and we simply call it *observation* o. Then, each observation o can occur after taking action a with a probability $P(o|a,h_t)$ which is the prediction for the test ao (azr) at history h_t. The agent receives a reward value r from

the environment, and the successor prediction vector ($P(Q|h_{t+1})$ is computed based on the resulting extended history *hazr*. In Chapter 3 we discussed that any dynamical system represented by POMDPs can be represented by a PSR model with number of core tests no larger than the number of states in the POMDP model. Considering this fact, planning in the space of reachable prediction vectors should intuitively be less complex than planning in the space of reachable POMDP belief states. To verify this we have a closer look at the planning space in a PSR model.

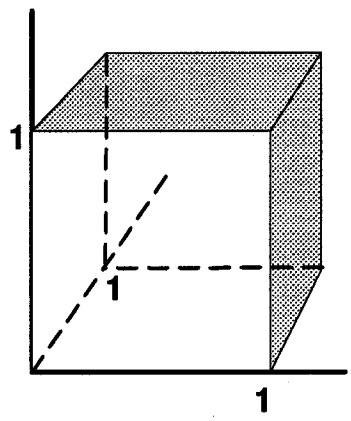


FIGURE 6.1. PSR space of prediction vectors for a problem with 3 core tests

As the dynamical system evolves in time, the prediction vector changes in a space of dimension $|Q|$ (Figure 6.1). But not every point in this space is a valid prediction vector (i.e. not every point corresponds to the prediction of core tests for a given a history). By the definition of a prediction vector as a vector of probabilities, the predictive state space (convex hull of prediction vectors) is somewhat different from the belief space of a POMDPs (as shown in Figure 2.2). This is due to the fact that a belief state is a probability vector whose elements sum up to 1. Therefore, for a belief vector of size n, we can write $b(s_n) = 1 - \sum_{i=1...n-1} b(s_i)$. This linear combination projects the beliefs into an $(n-1)$-dimensional space. The same projection is not applicable to prediction vectors.

If the linear PSR has the same dimensionality as the corresponding POMDP, then there is a one-to-one mapping between the space of the PSR's prediction vectors and the POMDP's space of reachable beliefs. Let U^Q be made of the outcome matrix of the PSR

core tests. The mapping from belief states to prediction vectors, after history h, is defined as: $f(b_h) = b_h^T U^Q = \rho$ is not reversible in general since U^Q has only $|Q|$ linearly independent columns. However, in the special case that the PSR model has the same dimension as the POMDP model (i.e. $|S| = |Q|$), the matrix U^Q is invertible, therefore the belief state b_h can be uniquely retrieved by the prediction vector as:

$$b_h = (P(Q|h)(U^Q)^{-1})^T \tag{6.5}$$

The number of distinct prediction vectors that can be experienced by the agent is at most the number of reachable belief states of the corresponding POMDP, given a fixed initial belief, although the space of prediction vectors looks different than the belief space. It must be noted that we considered above only the space of attainable points in both models, i.e. points for which there exists a history from a common initial state.

6.2.1. PSR Value Function

We can consider a policy tree at a given prediction vector, similarly to the case of policy trees for POMDP belief states (Figure 2.3). Such a tree defines an initial action taken under that policy and a one-step shorter policy tree for each observation. A POMDP policy tree for a deterministic π can reach only $|z|^l$ decision states in the next l time steps because it makes its decisions based solely on the observations made at each time step. Therefore, with the same analogy, a PSR policy tree contains at most $|o|^l$ paths of length l. Of course, the possibility of each branch in the tree depends on the prediction probability of the test it represents, given the prediction vector node it starts from. More precisely, given that initial action a of the policy π is taken at the initial prediction vector $\rho_0 = P(Q|h_0)$, policy tree π^o, that is shorter by one step and rooted at observation o has probability $p(o|a, \rho_0) = \rho_0^T m_{ao}$. The agent must select the best policy tree rooted at each decision point at each time step.

The value of a fixed policy tree π with a given initial action a, with respect to its initial decision node, ρ is:

$$\begin{aligned} V^{\pi}(\rho) &= R(\rho,a)+\gamma\sum_{o\in O}\rho^{T}m_{ao}V^{\pi^{o}}(\rho') \\ &= \rho^{T}\left(\eta_{a}+\gamma\sum_{o\in O}m_{ao}V^{\pi^{o}}(\rho')\right) \end{aligned}$$

where η_a is defined in Equation (6.3) and ρ' is the prediction vector obtained from ρ after a test ao. Policy trees can be used to search directly in the space of policies, similarly to policy iteration methods for POMDPs.

[James *et al.*, 2004] represented the PSR value function using policy tree machinery described above and developed a PSR value iteration method with incremental pruning algorithm. James et al. [2004] showed that the finite horizon value function over the space of prediction vectors is piecewise linear and convex. However, ensuring the validity of the obtained prediction vectors is tricky. Recall that each linear segment of the POMDP value function represents the optimal value of at least one belief state, and linear programming is used to find such a belief state. The validity of solution to this linear programming problem in POMDPs is done by a simple constraint $\sum_{i=1}^{n} b(s_i) = 1$. However, there's no clear and concrete constraint to check the validity of a found prediction vector corresponding to a linear segment of a PSR value function because prediction vectors are not probability distributions. Several constraints have been proposed in [James *et al.*, 2004]: each element of the prediction vector must be a probability, $0 \leq p_i \leq 1$; the components of the prediction vector corresponding to the same action sequence $a_1...a_n$ should sum to 1, $\sum_{o_1..o_n} p^T m_{a_1o_1..a_no_n} = 1$; each component of the prediction vector must correctly predict a core test, $p^T m_{q_i} = p_i$ for some $q_i \in Q$. However, these constraints do not seem to provide a sufficient condition for a valid prediction vector, occasionally causing problems in some domains [James *et al.*, 2004].

6.2.2. PSR Control and Look-ahead Search

The prediction vector summarizes all the information contained in the history of previous actions and observations. We use this information to evaluate different actions and to pick the best one. A simple idea is to look l steps into the future, and then pick the expected best action. The description of look-ahead search controllers [Barto *et al.*, 1995; Bonet *et al.*, 1997; Korf, 1990; Koenig and Simmons, 1998] can be slightly expanded to work for PSRs. As a first approach to the PSR control problem, we develop a look-ahead search to predict the expected utility of an action at a given prediction vector. At each time step, a look-ahead control policy chooses among feasible actions by envisioning their effects in the future and selecting the action which leads to the most promising predictive state. Following the same notation as in POMDPs, the value of taking action a at prediction

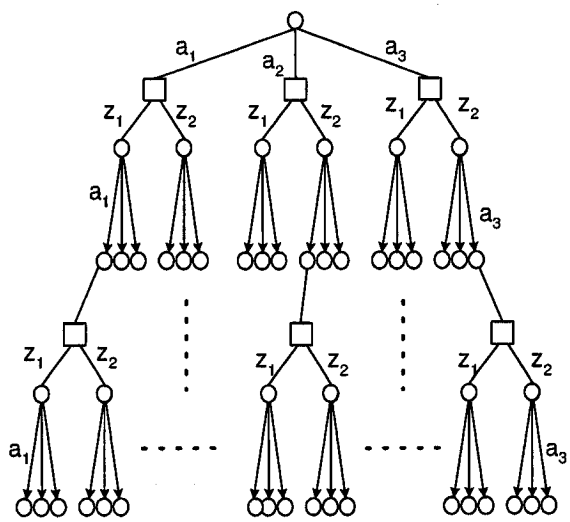


FIGURE 6.2. A lookahead tree with 3 actions and 2 observations for planning horizon of length 3

vector ρ can be defined as:

$$\mathbf{Q}(\rho,a) = R(\rho,a) + \gamma \sum_{o \in O} \rho^T m_{ao} V(\rho') \tag{6.6}$$

where $V(\rho')$ is the optimal value at the next prediction vector ρ'. Notice the difference between the above function $\mathbf{Q}$ and the set Q of core-tests. We assume that we are given the initial state of the system as the initial prediction vector ρ. The look-ahead controller

builds a look-ahead tree at each decision step. It estimates the values of each action using the above equation recursively and assigns the values of the leaves of the tree to minimum reward. This is a simplification, which works here because we assume that due to discounting, those values would have little impact anyway. The values at different levels are computed recursively. The depth of the tree, l, shows the horizon length, and the total size of the tree will be $(|A||O|)^l$. Therefore, in problems with large number of actions and observations the branching factor $|A||O|$ is high and makes this search very expensive.

At time t, we define the best action as:

$$a_t^* = \arg\ \max_a \mathbf{Q}(\rho_t, a) \tag{6.7}$$

Therefore, the policy of the agent can be determined by:

$$\pi^*(h,t) = a_t^* \tag{6.8}$$

This value may be cached and reused if the prediction vector ρ_t is revisited. The detailed algorithm is presented in algorithm 10. Techniques for limiting the search, e.g., pruning strategies [Dearden and Boutilier, 1994] or sparse sampling [Kearns *et al.*, 1999] can also be adopted for computational savings. Deeper look-ahead typically selects actions that lead to better outcomes. Of course deep forecasts are computationally expensive. Therefore, to keep the computation tractable, the look-ahead depth l cannot be very large. This is limiting, especially in problems for which the branching factor is very large. However, this is useful if a small horizon suffices still produces a good plan. The worst case running time of this algorithm is $O(|Q|^2(|A||o|)^l)$ where Q,A, and o present the set of core-test, actions, and observation-reward reward pairs respectively . Therefore the complexity of the algorithm is dominated by the horizon length and has less dependence on the size of the state space (the number of core-tests which gives the size of prediction vectors). This is similar to optimal planning algorithms for POMDPs. If there is some amount of compression in the PSR representation compared to the corresponding POMDP model ($|Q| < |S|$) then, potentially, there will be some savings in performing look-ahead using PSR prediction vectors. However, the exponential complexity of this technique in the horizon using

Algorithm 10 LookaheadControl(ρ, l)

INPUT Prediction vector ρ and the horizon length l
Initialize V to 0
if $l > 0$
for all actions a **do**
  $\mathbf{Q}(\rho, a) = 0$
  for all observations $o \in O$ **do**
    Compute the next prediction vector ρ' and $V(\rho')$ =LookaheadControl($\rho', l-1$)
  end for
  Compute the immediate reward $R(\rho, a)$ using Equation (6.4)
  Compute $\mathbf{Q}(\rho, a)$ using Equation (6.7)
end for
$a^* = \arg\max_a \mathbf{Q}(P(Q|h), a)$
$V^* = \max_a \mathbf{Q}(\rho, a)$
end if
Return a^* and V^*

both representations obscures this difference. In general, the two representations seem to be equally difficult to handle using exact methods. Our experimental results with this algorithm presented in Section 6.5, show that the PSR model as described in this chapter can be used in control problems to find comparable solutions to POMDPs. However, this algorithm and other similar approaches are not computationally efficient. In the next section we study the problem of approximate planning in PSRs using a more promising approach.

6.3. Approximate Planning with PSRs

Recall that approximate POMDP solution methods, such as point-based and grid-based techniques, rely on maintaining a value function only for a small subset of the belief space. Point- based methods generalize the plan over the entire continuous space of beliefs using the assumption that nearby points are likely to have nearby values. These methods differ from each other in the way they select the set $B = \{b_0, b_1, ..., b_m\}$ of reachable belief points. In this section we intend to study the applicability of the point-based framework to predictive state representations and we focus, in particular, on the point-based value iteration method of [Pineau *et al.*, 2003]. Recall that PBVI alternates between two phases: the value update for a current belief set B and the expansion of the belief set to get a better

approximation. Extending the algorithm to work for predictive state representation requires addressing these two key parts. For the first part, we modify the value update to work with prediction vectors in a straightforward way. However, deciding how to expand the set of reachable points similarly to POMDP-PBVI is less clear. In particular, the PBVI heuristics for belief point selection have all been developed based on specific criteria which rely on the fact that belief points are probability distributions, however, PSR prediction vectors do not form a probability space, as their components do not sum to 1. We discuss different selection heuristics for PSR prediction vectors and we develop an algorithm for PSRs.

Our results presented in Section 6.5 show that the PSR version of PBVI outperforms the original PBVI on POMDPs, especially when the PSR representation introduces some degree of compression compared to the corresponding POMDP.

6.3.1. Why Point-Based Planning for PSRs ?

As noted before, PSR prediction vectors can be more compact than POMDP belief states. Hence, PSRs can be viewed as a potential answer to the curse of dimensionality. At the same time, point-based methods have been designed to address the curse of history. Therefore, applying point-based techniques to PSRs can be viewed as addressing both the curse of dimensionality and the curse of history simultaneously. Attempts to achieve similar effects also exist in the POMDP literature. Poupart and Boutilier (2004) combine the value directed compression model [Poupart and Boutilier, 2000] with bounded policy iteration [Poupart and Boutilier, 2003a] to achieve a similar effect as we described in a model called VDCBPI [Poupart and Boutilier, 2004b]. Another related approach to predictive representations in approximate planning is the recent work on planning through *multiplicity automata* by Even-Dar et al. (2005). The authors convert the POMDP belief space of dimension n to an MDP of dimension $k \leq n$.

Another reason why we believe the point-based approach is a good choice for PSR planning is related to the problem of finding a value function for the space of prediction vectors. Point-based methods allow a discretization of this space based on reachable points. This discussion is based on the points that are reachable; hence, we believe that it may

provide a better representation of the value function than other approaches, always allowing to converge to valid predictions.

6.3.2. Metrics in the space of Prediction Vectors

In Chapter 4 we emphasized on the importance of the point selection strategy in the efficiency of point-based value iteration. In particular, we showed that the geometrical distance between belief state points can give a good idea of how distant the value of the points are from each other. Here we investigate whether the same metrics can be applied to prediction vector points.

We previously defined a belief point to prediction vector mapping. We use this mapping to relate pairwise distance between prediction vectors to the distance between reachable belief points.

THEOREM 10. *For any dynamical system, the POMDP to PSR mapping preserves the pairwise distance between points in the space of reachable beliefs, within a constant factor.*

PROOF. Given the same initial configurations, let b_0 be the initial belief of the POMDP representation, and b, b' two reachable beliefs from b_0 such that their distance in $d(b,b')$ is $\leq \varepsilon$.

$$\sum_i |b_i - b'_i| \leq \varepsilon$$

Since b, and b' are reachable from b_0, there exist histories h and h' to get to these points respectively. For $h = a_1o_1...a_no_n$:

$$b = \frac{b_0^T T^{a_1} O^{a_1 z_1} ... T^{a_n} O^{a_n z_n}}{b_0^T T^{a_1} O^{a_1 z_1} ... T^{a_n} O^{a_n z_n} 1_{|S|}} \tag{6.9}$$

Using the mapping from belief states to prediction vectors, for the two corresponding prediction vectors ρ and ρ' we have:

$$\begin{aligned} d(\rho,\rho') &= \| \rho - \rho' \|_1 \\ &= \| (b_h^T U^Q - b_{h'}^T U^Q \|_1 = \| (b_h^T - b_{h'}^T) U^Q \|_1 \text{ (PSR-POMDP mapping)} \\ &\leq \| b_h^T - b_{h'}^T \|_1 \| U \|_\infty \text{ (Holder inequality)} \\ &\leq \varepsilon n \text{ (elements of } U \text{ are probabilities)} \end{aligned}$$

□

The last inequality holds according to the definition of $\|\|_\infty$ for a matrix A as: $\| A \|_\infty = \max_i \sum_{j=1}^n |a_{ij}|$, and the fact that there are at most $n = |S|$ columns in U^Q. Therefore, we can use the heuristic for belief point expansions based on the distance between beliefs for the case of PSRs prediction vectors in the expansion phase of PBVI.

6.3.3. Extending PBVI to PSRs

The fact that the value function is piecewise linear and convex for PSR prediction vectors allows a good approximation of this function with a finite set of hyperplanes (α-vectors). The computation of the Bellman update equation can be performed on a set of prediction vector points $D = \{P(Q|h_1), P(Q|h_2), ..., P(Q|h_m)\}$ just as in the case of POMDPs. Hence the nth approximation V_n can be represented by a set of α-vectors $\Gamma_n = \{\alpha_1, ...\alpha_k\}$, such that each α_i corresponds to the best policy tree for at least one $P(Q|h_i) \in D$.

The backup operator is only performed on points in D based on the current value function V_n represented by Γ_n as:

$$V_{t+1}(\rho) = \arg \max_{a \in A} R(\rho,a) + \gamma(\sum_{\rho'} P(\rho'|\rho,a) V_t(\rho') \tag{6.10}$$

where $R(\rho,a)$ is the immediate reward of taking action a after the history h, given by the Equation (6.4), and $P(\rho'|\rho,a) = P(Q|hao)$ is the next prediction vector computed by Equation 3.8. Algorithm 11 details this approach. The set of vectors $\Gamma_t^{a,o}$ is used to find the estimates of values at the next prediction vector.

Algorithm 11 PSR point-based value backup(Γ_{t-1})

Input: previous value function estimate by Γ_{t-1}
for all $a \in A$ **do**
  for all $o \in O$ **do**
    $\Gamma_t^{a,o} \leftarrow \{\gamma M_{ao}\alpha | \alpha \in \Gamma_{t-1}\}$
  end for
end for
for all $\rho \in D$ **do**
  $\alpha^\rho \leftarrow \arg\ \max_{a \in A}(\eta_a + \sum_{o \in O} \max_{\alpha \in \Gamma_t^{ao}} \rho^T \alpha)$
end for
if $\alpha^\rho \notin \Gamma_t$
$\Gamma_t \leftarrow \Gamma_t \bigcup \alpha^\rho$
Return Γ_t

The number of candidate points to backup can be exponential if we intend to consider all reachable prediction vectors. Planning for horizon h can reach $O(|E|^h)$ points where E is the number of one-step extensions in the PSR model. This is $O((|A||Z|)^h)$ for POMDP-PBVI. The number of extensions can be estimated as $E = |A||O| = |A||Z||R|$, which exceeds $|A||Z|$. However, in practice not every observation-reward pair is possible for every action, and not every observation can generate all the rewards. This will be demonstrated in the experiments section. Using the theoretical results in the previous section we can derive an approximation bound for PSR-PVBI similar to POMDP-PVBI.

COROLLARY 1. *The error bounds for PBVI hold for the PSR-PBVI algorithm within a constant factor.*

PROOF. The proof follows from Theorem 9 and similar approach as in (Pineau et al., 2003) for POMDPs. Suppose the PSR-PBVI worst error is δ_D at a point $\rho' \notin D$. Suppose that instead of the correct optimal prediction vector α' at this point, the algorithm estimates the value using vector α which is the optimal α-vector for prediction vector $\rho \in D$:

$$\begin{aligned} \delta_D &= |(\alpha' - \alpha)(\rho - \rho')| \\ &\leq \max_{\rho'} \min_{\rho \in D} \| \rho - \rho' \|_\infty \\ &\leq \varepsilon n \end{aligned}$$

TABLE 6.1. Domain Description

Domain	$\|S\|$	$\|Q\|$	$\|A\|$	$\|Z\|$	$\|R\|$
4x4 grid	16	16	4	2	2
4x3 grid	11	11	4	6	4
Shuttle	8	7	3	5	3
Network	7	7	4	2	7

Domain	PSR reward	depth	optimal reward
4x4 grid	0.203	10	0.21
4x3 grid	0.08	7	0.11
Shuttle	1.98	8	2.02
Network	19.69	6	20.00

□

As we mentioned before, the error bound for PBVI is a loose bound; this bound is similarly loose. The running time of the PSR-PBVI algorithm is $O(|Q||A||O|)$ as opposed to the $O(|S||A||Z|)$ for the POMDP representation. Therefore, if the PSR provides a more compact representation, PSR-PBVI should be more efficient than the original POMDP-PBVI.

6.4. Experimental Results

Our experiments with PSR controllers consist of two parts. In the first part, experiments have been conducted on several test problems to compare the solution found by the PSR look-ahead approach with the POMDP optimal solution in each of these problems. In all the problems an initial prediction vector is specified. These problems are all small, in order to avoid the computational difficulty of generating PSR core tests and building the PSR model parameters in large domains. Another restriction is related to memory issues in building the lookahead trees for larger problems. Table 6.1 summarizes the descriptions of these problems. The performance of the PSR lookahead controller was examined based on the average reward per time step. The trajectories of length 1250 were used for each depth. We let the algorithm run for at most 8 hours on each domain and the results are based on the best reached depth. The depth of the lookahead tree used to get these solutions is shown in this table. The last column of the table shows the average reward per time step running

a POMDP optimal policy. As can be seen from this table, the lookahead approach using PSRs is very close to the optimal solution for all problems. In the case of the 4x3 grid we observed that a small horizon is not enough to reach the optimal solution.

The domains we used are summarized in Table 6.3. The coffee, 4x4 grid and shuttle problems have already been used in Chapter 4. In the second part of the experiments, we are interested to evaluate the performance of the PBVI-PSR algorithm. We use an additional domain which illustrates the advantage of the PSR model in point-based planning. Figure 6.3 depicts this example.

The orientation of the agent together with its physical position defines the states. Therefore, the POMDP representation of this domain consists of a 20-dimensional belief space. The actions are turning left, right and moving forward. The goal of the agent is to pass through the intersection, and it receives a reward of +1 just for doing that. The agent observes whether the square in front of it is blocked or unblocked, with some noise. The linear PSR representation uses only 5 core tests corresponding to the 5 states shown in the right panel of Figure 6.3. In the point set expansion phase of all algorithms we used

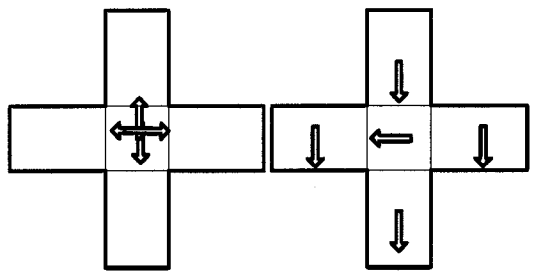


FIGURE 6.3. Robot-Intersection problem The state of the robot consists of the position in a grid together with the orientation in one of the four direction shown in the left. This makes the state space of the POMDP, $20-dimensional$. The figure in right depicts the state space of the corresponding PSR using only 5 distinctive experiences of the agent with the environment from the shown arrows.

forward simulation with explorative actions (SSEA) and the norm-1 metric for measuring distance between points in both belief space and prediction vector space. A set of 250 trajectories starting from a fixed given initial belief and following the policy generated by each method is used to evaluate the algorithms. The performance evaluation is conducted

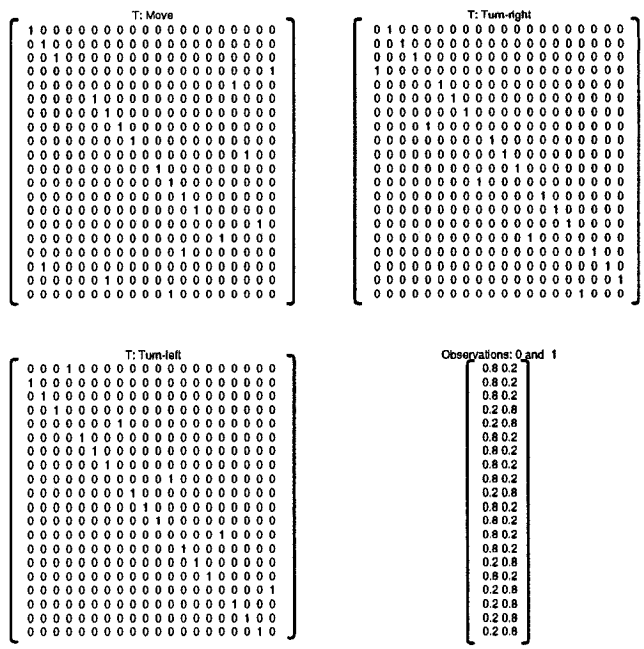


FIGURE 6.4. Description of the Robot-Intersection problem

in terms of averaging the discounted sum of the rewards obtained along these set of trajectories. The results in the table are averages over 10 such runs. We show the results of our

TABLE 6.2. Description of the Domains Used in PBVI-PSR experiments

Problem	$\|S\|$	$\|Q\|$	$\|A\|$	$\|O\|$	$\|R\|$	$\|E\|$
Robot-Intersection	20	5	3	2	2	7
Coffee	32	12	2	3	12	18
4x4-Grid	16	16	4	2	2	6
Shuttle	8	7	3	5	3	14

experiments with the PSR point-based algorithm compared to the reward-based POMDP (RPOMDP), and the original PBVI method. The RPOMDP model uses the same set of observations (observation and reward) as PSRs, but it does not provide any compression in the state space dimensions. Therefore, as we discussed in Chapter 5 it might increase the size of the observation space without providing more information. In all cases we performed 6 iterative expansions of the point set. The results in Table 6.4 show that the average discounted reward, which indicates the quality of the algorithms, the size of the set of points and the number of α-vectors representing the final estimated value function. The numbers

are averages over ten independent runs. The robot-intersection and the coffee domains are difficult to solve by POMDP exact methods and we could not get an optimal solution for these problems within a reasonable time.

All models provide similar solutions for all the problems except for the coffee problem. In this domain, since rewards are very informative, the planning space of PSR and RPOMDP is better, which explains the difference in results. It must be noted that the computation time is much lower for PSRs than RPOMDP because the PSR planning space is more compact.

Our experimental results in all cases show PSR-PBVI as the wining approach. This confirms our expectation that predictive state representations provide better time efficiency and speed up the progression of PBVI. The difference between the performance of these methods increases as the algorithms run for more expansions of their point set.

TABLE 6.3. Experimental results of PBVI for PSRs

Domain	Beliefs	α-vectors	Reward	Time (s)
Intersection				
POMDP	115	91	2.04 ± 0.05	6.6
RPOMDP	85	41	2.18 ± 0.02	3.3
PSR	85	27	2.18 ± 0.03	1.4
Coffee				
POMDP	78	18	-3.07 ± 0.84	3.4
RPOMDP	61	3	-2.26 ± 1.49	4.5
PSR	60	3	-2.16 ± 0.84	0.5
4x4-Grid				
POMDP	100	33	3.56 ± 0.06	3.8
RPOMDP	106	22	3.58 ± 0.1	5.8
PSR	103	17	3.63 ± 0.1	1.1
Shuttle				
POMDP	35	13	32.77 ± 0.1	1.1
RPOMDP	30	14	32.89 ± 0.1	2.6
PSR	30	12	32.87 ± 0.1	0.6

6.5. Summary and Conclusion

In this chapter we showed how the planning problems can be formulated for predictive state representation. We outlined the first exact planning algorithm for PSRs, which

we published in previous work. Our findings of PSR planning and the existing work on exact methods demonstrate that the intractability of finding an exact solution for POMDPs remains the same for the case of PSRs. However, using approximate planning, in particular point-based approach, seems to have a definite advantage in PSRs.

This combination targets the curse of dimensionality and the curse of history simultaneously. In this chapter, we also presented the first algorithm for approximate planning in PSRs, based on the PBVI approach in POMDPs. Our empirical evaluation of this algorithm shows considerable speed up compared to the point-based framework in POMDPs, while converging a solution that is at least as good.

Building the set of points for value iteration in point-based methods is very crucial to the quality of computed approximate plan. In the experiments we used stochastic simulation by explorative action. However, different schemes can be adopted, e.g. approaches similar to those explored in Chapter 4.

CHAPTER 7

Conclusions and Future Work

In this thesis we identified approaches which are likely to help with the two key issues involved in planning under uncertainty: the curse of dimensionality and the curse of history. The first approach is related to the design of more efficient POMDP approximation techniques. We proposed methods for focussing on small but essential parts of the belief space. The second approach builds on the predictive state representation as an alternative representation for planning. PSRs attempt to simplify, organize and make an elegant model capturing the useful information about the dynamics of the world. Predictive models are appealing because they connect the knowledge representation of the agent to observable experience with the environment. We identified cases in which PSRs provide a more compact model than POMDPs, by exploiting regularities in the state dynamics. The PSR model also encodes reward information in the state representation, which leads to less uncertainty and more precise abstractions. We reformulated the optimal control problem using PSRs and extended two POMDP planning algorithms for PSRs.

We now re-visit the main contributions of the thesis and highlight avenues for future work.

7.1. Summary

Throughout chapter 4 we presented a collection of results for belief point selection in the class of point based POMDP approximation methods. We illustrated the importance of

the point selection problem, and proposed new heuristics that try to solve this problem in a more efficient way within the point-based value iteration (PBVI) algorithm. We studied a number of criteria designed to improve the selection of candidate points. We defined the concept of core beliefs, a linearly independent set of belief states that spans the whole space of reachable beliefs in a POMDP. Considering the core beliefs in point-based methods guarantees a tighter worst case error bound than standard PBVI [Izadi *et al.*, 2005]. We studied two additional heuristics designed to improve the PBVI algorithm. The first of these methods is based on the reachability analysis of the POMDP belief space. This approach relies on prioritization of beliefs based on the degree of reachability from the given initial belief state [Izadi *et al.*, 2006]. We established the theoretical soundness of this approach in Chapter 4. The second approach is motivated by the observation that the exploration-exploitation trade off in the space of beliefs has a great influence on the precision of value function and the time needed to estimate it [Izadi and Precup, 2006]. We suggested a novel approach that uses a distance threshold parameter. We presented empirical evaluations illustrating how PBVI can be influenced by these approaches. While each heuristic can be especially useful for a particular kind of system, our last approach works generally better on all types of domains.

We have pointed out the characteristics of the PSR model which are useful for structure exploitation in chapter 5, and showed that the PSR core tests discovery algorithm automatically generates a good state aggregation on the underlying states of the corresponding POMDP. We addressed the issue of modeling symmetries in the dynamics of the system through predictive state representations. The notion of state dependency as a property of the underlying states of a POMDP was developed for the first time. This is the basis of a state abstraction method. However, we emphasized that for this abstraction to respect the value function and not to aggregate states with potentially different values, rewards should be encoded in the representation of the beliefs. Reward involvement in the definition of belief states can also be useful for planning purposes. We demonstrated this fact with experiments using the new formulation of beliefs. We were able to get good results for a class of problems in which rewards are informative.

We studied the applicability and usefulness of PSRs in control problems; we designed two planning algorithms in Chapter 6, based on existing POMDP solution methods. We provided both the theoretical background for using PSRs in planning and empirical evidence that PSRs are particularly well-suited for approximate planning using point based algorithms. The algorithms described here preserve the worst-case error bounds for POMDPs, but can have a substantial impact on empirical performance. The empirical study in Chapter 6 demonstrates that the PSR representation with the PBVI algorithm improves slightly the solution quality over the best POMDP planning methods but is significantly faster. The impact on quality is more pronounced for problems in which the PSR representation provides more compression.

7.2. Future Extensions

Real world applications have large or continuous state spaces. However, these environments usually exhibit a lot of structure in their dynamics and in their states. We analyzed the ability of linear PSRs to discover a particular type of structure in a system. However, linear PSRs with the current notion of tests is limited, as we showed in Chapter 5. Therefore, it would be desirable to explore the type of structure that other variations of PSR can exploit. For instance, EPSRs have an interesting test definition, similar to closed loop policies, options, and hierarchical actions. Hierarchical structures have been the focus of substantial work in the reinforcement literature [Singh, 1992a; Dietterich, 1998; Pineau and Thrun, 2001; Barto and Mahadevan, 2003; Bakker and Schmidhuber, 2004; Dietterich, 2000a]. Options, as temporally abstract actions, have efficient learning and planning algorithms in reinforcement learning with temporal abstraction [Stolle and Precup, 2002; Sutton *et al.*, 1999c; Precup *et al.*, 1998; Precup and Sutton, 1998]. Augmenting the predictive state representation with options looks encouraging; [Wolfe and Singh, 2006] can be considered an early step in this direction. Our results in Chapter 5 can be extended for analyzing EPSRs as well.

Another extension of this work is to use the motivation behind the suggested heuristics in Chapter 4 to design more sophisticated algorithms. Most of the computation time in

point-based algorithms is spent on the point-based value update phase. In our experiments we observed that a much smaller subset of points are essential in representing the optimal value function. Although it is difficult to clarify which points and which transitions are important without explicitly solving the problem, one important direction would be to investigate the appropriateness of other similarity metrics to reduce the problem. There is still a lack of quantitative analysis for the precision of PBVI approximations. Techniques developed in the thesis and previously suggested approaches try to reduce the approximation error. However we still seek theoretical results to answer questions such as: how many points suffice to have an approximation of the value function within at most ε distance from the optimal solution? While this returns in part to the complexity of the optimal value function, it is still not clear how to characterize this complexity based on the dynamics of the model. Some work from mathematical analysis [Shapiro *et al.*, 2000] seems relevant and inspiring.

The structural analysis of predictive representations given in the thesis can motivate the development of other variation of tests. For example, we showed in Chapter 5 that symmetrical structure with respect to the space of state-action pairs cannot always be detected by the linear PSR. It would be interesting to have tests that can capture this property. Representations and algorithms that exploit additional structure can be of great practical value.

The predictive representation does not rely on a specific physical layout of an unobservable environment, so it has the potential of being useful for fast adaptation to a new similar environment. From the perspective of generalizing across tasks, PSRs can be quite useful. We plan to investigate this further in the future.

In light of the difficulty of the discovery problem for core tests, it is natural to look for approximation algorithms which generate more appropriate core tests for planning purposes. This requires a precise definition of a metric which best describes the usefulness of different core tests in value function computation. A simple example can be a distance metric which characterizes core tests with the mass of possible prediction vectors they represent. This identifies the potential error that missing a particular core tests may cause.

Another possible direction is to consider restrictions on the representational level of tests (e.g. limiting the size of core tests). These types of analysis have been suggested or carried out in the literature for abstraction with other representations [Poupart and Boutilier, 2003b; Hundt *et al.*, 2006; Sutton and Tanner, 2004]. Finding the appropriate restrictions for PSRs is an open question.

REFERENCES

[Aberdeen and Baxter, 2002] Douglas Aberdeen and Jonathan Baxter. Scalable internal-state policy-gradient methods for pomdps. In *International Conference on Machine Learning (ICML-02)*, pages 3–10, 2002.

[Aberdeen, 2005] Douglas Aberdeen. Policy-gradient methods for planning. In *Neural Information Processing Systems (NIPS-05)*, 2005.

[Astrom, 1965] K. J. Astrom. Optimal control of markov decision processes with incomplete state estimation. *Mathematical Analysis and Applications*, 10:174–205, 1965.

[Bakker and Schmidhuber, 2003] B. Bakker and J. Schmidhuber. Hierarchical reinforcement learning based on sub-goal discovery and sub-policy specialization: First experiments with the hassle algorithm, 2003.

[Bakker and Schmidhuber, 2004] B. Bakker and J. Schmidhuber. Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization. In *The 8-th Conference on Intelligent Autonomous Systems, IAS-8, Amsterdam, The Netherlands*, pages 438–445, 2004.

[Barto and Mahadevan, 2003] A. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications*, 13:41–77, 2003.

[Barto *et al.*, 1995] A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1-2):81–138, 1995.

[Beimel *et al.*, 1996] A. Beimel, F. Bergadano, N. H. Bshouty, E. Kushilevitz, and S. Varricchio. On the applications of multiplicity automata in learning. In *IEEE Symposium on Foundations of Computer Science*, pages 349–358, 1996.

[Bellman, 1957] R. E. Bellman. *Dynamic programming*. Princeton University Press, 1957.

[Bernstein *et al.*, 2005] D. S. Bernstein, E. A. Hansen, and S. Zilberstein. Bounded policy iteration for decentralized pomdps. In *International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1287–1292, 2005.

[Bertsekas, 1987] D. P. Bertsekas. *Dynamic programming: deterministic and stochastic models*. Prentice-Hall, Inc., 1987.

[Bonet *et al.*, 1997] B. Bonet, G. Loerincs, and H. Geffner. A robust and fast action selection mechanism for planning. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97)*, pages 714–719, 1997.

[Boutilier and Poole, 1996] C. Boutilier and D. Poole. Computing optimal policies for partially observable markov decision process using compact representations. In *National Conference on Artificial Intelligence (AAAI-96)*, pages 1168–1175, 1996.

[Boutilier *et al.*, 2005] C. Boutilier, R. Patrscu, P. Poupart, and D. Schuurmans. Regret-based utility elicitation in constraint-based decision problems. In *The International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 929–934, 2005.

[Boutilier, 2002] C. Boutilier. A pomdp formulation of preference elicitation problems. In *Eighteenth national conference on Artificial intelligence (AAAI-02)*, pages 239–246, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[Boyen and Koller, 1998] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Conference on Uncertainty in Artificial Intelligence (UAI98)*, pages 33–42, 1998.

[Brafman, 1997] R. Brafman. A heuristic variable grid solution method for pomdps. In *The Fourteenth National Conference on Artificial Intelligence(AAAI-97)*, pages 727–733, 1997.

[Braziunas and Boutilier, 2004] D. Braziunas and C. Boutilier. Stochastic local search for POMDP controllers. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, pages 690–696, San Jose, CA, 2004.

[Braziunas, 2003a] D. Braziunas. Pomdp solution methods: a survey. Technical report, Department of Computer Science, University of Toronto, 2003.

[Braziunas, 2003b] D. Braziunas. Stochastic local search for pomdp controllers. Master's thesis, University of Toronto, Toronto, 2003.

[Braziunas, 2006] Darius Braziunas. Computational approaches to preference elicitation. Technical report, Department of Computer Science, University of Toronto, 2006.

[Cassandra *et al.*, 1994] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, volume 2, pages 1023–1028, Seattle, Washington, USA, 1994. AAAI Press/MIT Press.

[Cassandra *et al.*, 1996] A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien. Acting under uncertainty: discrete bayesian models for mobile robot navigation. In *IEEE/RSJ International Conference on Intelligent Robot and Systems*, pages 963–972, 1996.

[Cassandra *et al.*, 1997] A. Cassandra, M. L. Littman, and N. L. Zhang. Incremental pruning: A simple, fast, exact method for partially observable markov decision processes. In *13th Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 54–61, 1997.

[Cassandra, 1995] A. Cassandra. Tony's pomdp repository, 1995.

[Cassandra, 1998a] A. Cassandra. *Exact and Approximate Algorithms for Partially Observable Markov Decission Processes*. PhD thesis, Brown University, 1998.

[Cassandra, 1998b] A. Cassandra. A survey of pomdp applications. In *Working Notes of AAAI, Fall Symposium on Planning with Partially Observable Markov Decision Processes*, pages 17–24, 1998.

[Cheng, 1988a] H. Cheng. *Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, University of British Columbia, 1988.

[Cheng, 1988b] H. T. Cheng. *Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, University of British Columbia, Vancouver, 1988.

[Chong *et al.*, 2004] E. K. P. Chong, U. Savagaonkar, and R. Givan. Online pricing for bandwidth provisioning in multi-class networks. In *Workshop on Economics of Communication Networks*, 2004.

[Chrisman, 1992] Lonnie Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *National Conference on Artificial Intelligence*, pages 183–188, 1992.

[Dayan and Hinton, 1993] P. Dayan and G. E. Hinton. Feudal reinforcement learning. In C. L. Giles, S. J. Hanson, and J. D. Cowan, editors, *Advances in Neural Information Processing Systems 5, Proceedings of the IEEE Conference in Denver (to appear)*, pages 271–278, San Mateo, CA, 1993. Morgan Kaufmann.

[Dayan, 1993] P. Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5:613–624, 1993.

[Dearden and Boutilier, 1994] R. Dearden and C. Boutilier. Integrating planning and execution in stochastic domains. In *Proceedings of the AAAI Spring Symposium on Decision Theoretic Planning*, pages 55–61, Stanford, CA, 1994.

[Dearden and Boutilier, 1997] R. Dearden and C. Boutilier. Abstraction and approximate decision theoretic planning. *Artificial Intelligence*, 89(1):219–283, 1997.

[Desharnais *et al.*, 2006] J. Desharnais, F. Laviolette, K. Priya, D. Moturu, and S. Zhioua. Trace equivalence characterization through reinforcement learning. In *Canadian Conference on AI*, pages 371–382, 2006.

[Dietterich, 1998] T. G. Dietterich. The MAXQ method for hierarchical reinforcement learning. In *International Conference on Machine Learning (ICML-98)*, pages 118–126. Morgan Kaufmann, San Francisco, CA, 1998.

[Dietterich, 2000a] T. Dietterich. State abstraction in maxq hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems 12*, pages 994–1000, 2000.

[Dietterich, 2000b] T. G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–313, 2000.

[Drake, 1962] A. Drake. *Observation of a Markov Process Through a Noisy Channel.* PhD thesis, Massachusetts Institute of Technology, 1962.

[Dutech, 2000] A. Dutech. Solving pomdps using selected past events. In *The Fourteenth European Conference on Artificial Intelligence*, pages 281–285, 2000.

[Even-Dar *et al.*, 2005] E. Even-Dar, S. M. Kakade, and Y. Mansour. Planning in pomdps using multiplicity automata. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 185–19, Arlington, Virginia, 2005.

[Feng and Hansen, 2001] Z. Feng and E. Hansen. Approximate planning for factored pomdps. In *European Conference on Planning*, 2001.

[Ferns *et al.*, 2005] N. F. Ferns, P. Panangaden, and D. Precup. Metrics for markov decision processes with infinite state spaces. In *21st Conference on Uncertainty in Artificial Intelligence*, page 201, 2005.

[Giunchiglia and Walsh, 1992] F. Giunchiglia and T. Walsh. A theory of abstraction. *Artificial Intelligence*, 57(2-3):323–390, 1992.

[Givan *et al.*, 2003] R. Givan, T. Dean, and M. Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1-2):163–223, 2003.

[Guestrin *et al.*, 2001] C. Guestrin, D. Koller, and R. Parr. Solving factored POMDPs with linear value functions. In *Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01) workshop on Planning under Uncertainty and Incomplete Information*, pages 67–75, Seattle, Washington, August 2001.

[Hansen and Feng, 2000] E. Hansen and Z. Feng. Dynamic programming for pomdp using a factored state representation. In *International Conference on Artificial Intelligence Planning and Scheduling*, pages 130–139, 2000.

[Hansen and Zhou, 2003a] E. A. Hansen and R. Zhou. Synthesis of hierarchical finite-state controllers for pomdps. In *13th International Conference on Automated Planning and Scheduling (ICAPS-03)*, pages 113–122, 2003.

[Hansen and Zhou, 2003b] E. A. Hansen and R. Zhou. Synthesis of hierarchical finite-state controllers for pomdps. In *ICAPS*, pages 113–122, 2003.

[Hansen, 1998] E. A. Hansen. An improved policy iteration algorithm for partially observable mdps. In *Tenth Conference on Advances in Neural Information Processing Systems, NIPS*, pages 1015–1021, Cambridge, MA, USA, 1998. MIT Press.

[Hauskrecht, 1997] M. Hauskrecht. Incremental methods for computing bounds in partially observable markov decision processes. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 734–739, 1997.

[Hauskrecht, 2000a] M. Hauskrecht. Value-function approximations for partially observable markov decision. *Journal of Artificial Intelligence Research*, pages 33–94, 2000.

[Hauskrecht, 2000b] M. Hauskrecht. Value-function approximations for partially observable markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000.

[Hoey and Poupart, 2005] J. Hoey and P. Poupart. Solving pomdps with continuous or large discrete observation spaces, 2005.

[Hopcroft and Ullman, 1979] J.E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Massachusetts, 1979.

[Howard, 1960] R. A. Howard. *Dynamic programming and Markov processes.* MIT Press, Cambridge, 1960.

[Hundt *et al.*, 2006] C. Hundt, P. Panangaden, J. Pineau, and D. Precup. Representing systems with hidden state. In *National Conference on Artificial Intelligence (AAAI-06)*, 2006.

[Izadi and Precup, 2003] M. T. Izadi and D. Precup. A planning algorithm for predictive state representation. In *The 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 1520–1521, Acapulco, Mexico, 2003.

[Izadi and Precup, 2005a] M. T. Izadi and D. Precup. Model minimization by linear psr. In *The 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1749–1750, 2005.

[Izadi and Precup, 2005b] M. T. Izadi and D. Precup. Using rewards for belief state updates in partially observable markov decision processes. In *European Conference on Machine Learning ECML05*, pages 593–600, 2005.

[Izadi and Precup, 2006] M.T. Izadi and D. Precup. Exploration in pomdp belief space and its impact on value iteration approximation. In *European Conference on Artificial Intelligence. Workshop on Planning, Learning, and Monitoring with Uncertainty and Dynamic Worlds*, 2006.

[Izadi *et al.*, 2005] M. T. Izadi, A. V. Rajwade, and D. Precup. Using core beliefs for point-based value iteration. In *The 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1751–1753, 2005.

[Izadi *et al.*, 2006] M. T. Izadi, D. Precup, and D. Azar. Belief selection in point-based planning algorithms for pomdps. In *Canadian Conference on AI*, pages 383–394, 2006.

[Jaeger, 1998] H. Jaeger. Discrete-time discrete-valued observable operator models: A tutorial, 1998.

[Jaeger, 2000] H. Jaeger. Observable operator models for discrete stochastic time series. *Neural Computation*, 12(6):1371–1398, 2000.

[James and Singh, 2004a] M. R. James and S. Singh. Learning and discovery of predictive state representations in dynamical systems with reset. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, pages 53–59, 2004.

[James and Singh, 2004b] M. R. James and S. Singh. Learning and discovery of predictive state representations in dynamical systems with reset. In *Proceedings of the twenty-first international conference on Machine learning (ICML-04)*, page 53, New York, NY, USA, 2004. ACM Press.

[James and Singh, 2005] M. R. James and S. P. Singh. Planning in models that combine memory with predictive representations of state. In *The National Conference on Artificial Intelligence (AAAI-05)*, pages 987–992, 2005.

[James *et al.*, 2004] M. James, S. Singh, and M. Littman. Planning with predictive state representation. In *International Conference on Machine Learning and Applications (ICMLA-04)*, 2004.

[James *et al.*, 2005] M. R. James, B. Wolfe, and S. P. Singh. Combining memory and landmarks with predictive state representations. In *International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 734–739, 2005.

[James, 2005] M.R. James. *Using predictions for planning and modeling in stochastic environments*. PhD thesis, The University of Michigan, 2005.

[Jaulmes *et al.*, 2005] Robin Jaulmes, Joelle Pineau, and Doina Precup. Active learning in partially observable markov decision processes. In *ECML*, pages 601–608, 2005.

[Jong and Stone, 2005] N.K. Jong and P. Stone. State abstraction discovery from irrelevant state variables. In *International Joint Conference on Artificial Intelligence (IJCAI05)*, pages 752–757, 2005.

[Jou and Smolka, 1990] C. C. Jou and S. A. Smolka. Equivalence congruence and complete maximization for probabilistic processes. In J.C.M. Baeten and J.W. Klop, editors, *CONCUR 90 First International Conference on Concurrency Theory*, number 458 in Lecture Notes In Computer Science. Springer-Verlag, 1990.

[Kaelbling *et al.*, 1998] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101, 1998.

[Kaelbling, 1993] L.P. Kaelbling. Hierarchical reinforcement learning: Preliminary results. *Neural Computations.*, 6(6):1185–1201, 1993.

[Kearns *et al.*, 1999] M. Kearns, Y. Mansour, and A. Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *International Joint Conference on Artificial Intelligence (IJCAI99)*, pages 1324–1231, 1999.

[Koenig and Simmons, 1998] S. Koenig and R. G. Simmons. Xavier: A robot navigation architecture based on partially observable markov decision process models. In *Artificial Intelligence Based Mobile Robotics:Case Studies of Successful Robot Systems*, pages 91–122, 1998.

[Korf, 1990] R. E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211, 1990.

[Larson and Skou, 1994] K. G. Larson and A. Skou. Bisimulation through probabilistic testing. *Information and Computing*, 1(128):1–28, 1994.

[Li and Littman, 2005] L. Li and M. L. Littman. Lazy approximation for solving continuous finite-horizon mdps. In *National Conference on Artificial Intelligence AAAI-05*, pages 1175–1180, 2005.

[Li *et al.*, 2006] L. Li, T. Walsh, and M. L. Littman. Towards a unified theory of state abstraction for mdps. In *The 9th international Symposium on Artificial Intelligence*, 2006.

[Littman *et al.*, 1995a] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Efficient dynamic programming updates in partially observable markov decision processes. Technical report, Brown University, 1995.

[Littman *et al.*, 1995b] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. In Armand Prieditis and Stuart Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 362–370, San Francisco, CA, USA, 1995. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[Littman *et al.*, 2001] M. Littman, R. Sutton, and S. Singh. Predictive representations of state, 2001.

[Littman, 1994] M. L. Littman. The witness algorithm: solving partially observable markov decision processes. Technical Report Technical Report CS-94-40, Brown University, 1994.

[Littman, 1997] M. L. Littman. Probabilistic propositional planning: representations and complexity. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97)*, pages 748–761, 1997.

[Lovejoy, 1991a] W. S. Lovejoy. Computationally feasible bounds for partially observable markov decision processes. *Operations Research*, 39:175–192, 1991.

[Lovejoy, 1991b] W. S. Lovejoy. A survey of algorithmic methods for partially observed markov decision processes. *Ann. Operations Research*, 28(1-4):47–66, 1991.

[Madani *et al.*, 2003] O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*, 147(1-2):5–34, 2003.

[Mannor *et al.*, 2004] Shie Mannor, Ishai Menache, Amit Hoze, and Uri Klein. Dynamic abstraction in reinforcement learning via clustering. In *International Conference on Machine Learning*, page 71, 2004.

[McCallum, 1993] R. A. McCallum. Overcoming incomplete perception with util distinction memory. In *International Conference on Machine Learning*, pages 190–196, 1993.

[McCallum, 1995a] A. McCallum. Instance-based utile distinctions for reinforcement learning with hidden state. In *International Conference on Machine Learning*, pages 387–395, 1995.

[McCallum, 1995b] A. McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, 1995.

[McCracken and Bowling, 2006] P. McCracken and M. Bowling. Online discovery and learning of predictive state representations. In *Advances in Neural Information Processing Systems (NIPS-18)*, pages 875–882. MIT Press, Cambridge, MA, 2006.

[McGovern *et al.*, 2001] A. McGovern, , and A. G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *International Conference on Machine Learning (ICML-01)*, pages 361–368, 2001.

[Meuleau *et al.*, 1999] N. Meuleau, L. Peshkin, K. Kim, and L. P. Kaelbling. Learning finite-state controllers for partially observable environments. In *Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 427–436, 1999.

[Mitchell, 2003] M. Mitchell. Using markov-k memory for problems with hidden state, 2003.

[Monahan, 1982] G. E. Monahan. A survey of partially observable markov decision processes: Theory, methods and algorithms. *Management Science*, 28:1–16, 1982.

[Montemerlo *et al.*, 2002] M. Montemerlo, J. Pineau, N. Roy, S. Thrun, and V. Verma. Experiences with a mobile robot guide for the elderly. In *Eighteenth National Conference on Artificial Intelligence*, pages 587–592, 2002.

[Mundhenk *et al.*, 1997] M. Mundhenk, J. Goldsmith, and E. Allender. The complexity of policy-evaluation for finite-horizon partially observable markov decision processes. In *Proceedings of 22nd Symposium on Mathematical Foundations of Computer Science (published in Lecture Notes in Computer Science)*, pages 129–138. Springer-Verlag, 1997.

[Murphy, 2000] K. Murphy. A survey of pomdp solution techniques. Technical report, U.C. Berkeley, 2000.

[Ng and Jordan, 2000] A.Y. Ng and M. Jordan. Pegasus: A policy search method for large mdps and pomdps. In *Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 406–415, 2000.

[Nourbakhsh *et al.*, 1994] I. Nourbakhsh, R. Powers, and S. Birchfield. Dervish an office-navigating robot. Artificial Intelligence Magazine, 1994.

[Paek and Horvitz, 2000] T. Paek and E. Horvitz. Conversation as action under uncertainty. In *Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, pages 455–464, 2000.

[Parr and Russell, 1995] R. Parr and S. Russell. Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995.

[Pineau and Gordon, 2005] J. Pineau and G. Gordon. Pomdp planning for robust robot control. In *International Symposium on Robotics Research (ISRR)*, 2005.

[Pineau and Thrun, 2001] J. Pineau and S. Thrun. Hierarchical pomdp decomposition for a conversational robot, 2001.

[Pineau and Thrun, 2002] J. Pineau and S. Thrun. An integrated approach to hierarchy and abstraction for pomdps. Technical Report CMU-RI-TR-02-21, Carnegie Mellon University. Robotics Institute., 2002.

[Pineau *et al.*, 2003] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithms for pomdps. In *International Joint Conference on Artificial Intelligence(IJCAI03)*, pages 1025–1032, 2003.

[Pineau *et al.*, 2006] J. Pineau, G. Gordon, and S. Thrun. Anytime point-based approximations for large pomdps. *Journal of Artificial Intelligence Research*, 27:335–380, 2006.

[Pineau, 2004] Joelle Pineau. *Tractable Planning Under Uncertainty: Exploiting Structure*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2004.

[Platzman, 1997] L. K. Platzman. *Finite-memory estimation and control of finite probabilistic systems*. Phd thesis, Massachusetts Institute of Technology, 1997.

[Porta *et al.*, 2005] Josep M. Porta, Matthijs T. J. Spaan, and Nikos Vlassis. Robot planning in partially observable continuous domains. In *Proceedings of Robotics: Science and Systems*, 2005.

[Poupart and Boutilier, 2000] P. Poupart and C. Boutilier. Value-directed belief state approximation for pomdps. In *Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-2000*, pages 497–506, 2000.

[Poupart and Boutilier, 2003a] P. Poupart and C. Boutilier. Bounded finite state controllers, 2003.

[Poupart and Boutilier, 2003b] P. Poupart and C. Boutilier. Value-directed compression for pomdps. In *Advances in Neural Information Processing Systems (NIPS-03)*, pages 1547–1554, 2003.

[Poupart and Boutilier, 2004a] P. Poupart and C. Boutilier. Bounded finite state controller. In *Advances in Neural Information Processing Systems (NIPS-04)*, 2004.

[Poupart and Boutilier, 2004b] P. Poupart and C. Boutilier. Vdcbpi: an approximate scalable algorithm for large scale pomdps. In *In Advances in Neural Information Processing Systems 17 (NIPS-04)*, pages 1081–1088, 2004.

[Precup and Sutton, 1998] D. Precup and R. S. Sutton. Multi-time models for temporally abstract planning. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10, pages 1050–1056. The MIT Press, 1998.

[Precup *et al.*, 1998] D. Precup, R. S. Sutton, and S. P. Singh. Theoretical results on reinforcement learning with temporally abstract options. In *Proceedings of the 10th European Conference on Machine Learning (ECML-98)*, pages 382–393, London, UK, 1998. Springer-Verlag.

[Puterman, 1994] M. L. Puterman. *Markov Decision Process: Discrete stochastic dynamic programming*. John Wiley and Sons Inc., 1994.

[Rafols *et al.*, 2005] E. J. Rafols, M. B. Ring, R. S. Sutton, and B. Tanner. Using predictive representations to improve generalization in reinforcement learning. In *IJCAI-05*, pages 835–840, 2005.

[Ravindran and Barto, 2002] B. Ravindran and A. G. Barto. Model minimization in hierarchical reinforcement learning. In *International Symposium on Abstraction, Reformulation and Approximation (SARA-02)*, volume 2371 of *LNCS*, pages 196–211, 2002.

[Rivest and Schapire, 1994] R. L. Rivest and R. E. Schapire. Diversity-based inference of finite automata. *J. ACM*, 41(3):555–589, 1994.

[Rodriguez *et al.*, 1999] A. Rodriguez, R. Parr, and D. Koller. Reinforcement learning using approximate belief state. In *NIPS99*, pages 1036–1042, 1999.

[Roy *et al.*, 2000] N. Roy, J. Pineau, and S. Thrun. Spoken dialog management for robots. In *Proceedings of the ACL 2000*, October 2000.

[Roy *et al.*, 2005] Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. Finding approximate pomdp solutions through belief compression. *Journal of Artificial Intelligence Research*, 23:1–40, 2005.

[Roy, 2003] N. Roy. *Finding approximate POMDP solutions through belief compression.* PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, September 2003.

[Rudary and Singh, 2004] M. Rudary and S. Singh. A nonlinear predictive state representation. In *Advances in Neural Information Processing Systems 16*, 2004.

[Rudary and Singh, 2006] M. Rudary and S. Singh. Predictive linear-gaussian models of controlled stochastic dynamical systems. In *The 23rd International Conference on Machine Learning (ICML-06)*, pages 777–784, 2006.

[Rudary *et al.*, 2005] M. Rudary, S. Singh, and D. Wingate. Predictive linear-gaussian models of stochastic dynamical systems. In *The 21th Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 501–506, Arlington, Virginia, 2005. AUAI Press.

[Shani, 2004] G. Shani. A survey of model-based and model-free methods for resolving perceptual aliasing. Technical report, The Ben-Gurion University in the Negev, 2004.

[Shapiro *et al.*, 2000] A. Shapiro, T. Homem-de-Mello, and J. Kim. Conditioning of stochastic programs. Math. Program Series, 2000.

[Shlitzenberger, 1961] M.P Shlitzenberger. On the definition of a family of automata. *Information and Control.*, 4 (2-3):245–270, 1961.

[Simsek *et al.*, 2005] Ozgur Simsek, Alicia P. Wolfe, and Andrew G. Barto. Identifying useful sub-goals in reinforcement learning by local graph partitioning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 816–823, 2005.

[Singh *et al.*, 2003] S. Singh, M. Littman, N. Jong, D. Pardoe, and P. Stone. Learning predictive state representations, 2003.

[Singh *et al.*, 2004] S. Singh, M. R. James, and M. R. Rudary. Predictive state representations: a new theory for modeling dynamical systems. In *AUAI '04: Proceedings of the*

20th conference on Uncertainty in artificial intelligence, pages 512–519, Arlington, Virginia, United States, 2004. AUAI Press.

[Singh, 1992a] S. P. Singh. Reinforcement learning with a hierarchy of abstract models. In *National Conference on Artificial Intelligence*, pages 202–207, 1992.

[Singh, 1992b] Satinder P. Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8:323–339, 1992.

[Smallwood and Sondik, 1973] R. D. Smallwood and E. J. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

[Smith and Simmons, 2004] T. Smith and R. Simmons. *Heuristic search value iteration for POMDPs*. Proceedings of UAI, 2004.

[Smith and Simmons, 2005] Trey Smith and Reid Simmons. Point-based pomdp algorithms: Improved analysis and implementation. In *Uncertainty in Artificial Intelligence (UAI-05)*, pages 542–555, July 2005.

[Sondik, 1971] E. J. Sondik. *The optimal control of Partially Observable Markov Processes*. Ph.D. thesis, Stanford University, 1971.

[Spaan and Vlassis, 2005] M. T. J. Spaan and N. Vlassis. *Perseus: Randomized point-base value iteration for POMDPs*. Journal of Artificial Intelligence Research, 2005.

[Spaan, 2006] Matthijs T. J. Spaan. *Approximate planning under uncertainty in partially observable environments*. PhD thesis, Universiteit van Amsterdam, 2006.

[Stolle and Precup, 2002] M. Stolle and D. Precup. Learning options in reinforcement learning. In *Proceedings of the 5th International Symposium on Abstraction, Reformulation and Approximation*, pages 212–223, London, UK, 2002. Springer-Verlag.

[Sutton and Tanner, 2004] R. S. Sutton and B. Tanner. Temporal-difference networks. In *Neural Information Processing Systems (NIPS-05)*, 2004.

[Sutton *et al.*, 1999a] R. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. Technical report, ATT Research Labs,, 1999.

[Sutton *et al.*, 1999b] R. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

[Sutton *et al.*, 1999c] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: a framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

[Sutton, 1988] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[Tambe and the TEAMCORE group University of Southern California, 2005] Milind Tambe and the TEAMCORE group University of Southern California. Resolving conflicts about teamwork: hybrids to the rescue. In *International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS05)*, 2005.

[Tanner and Sutton, 2005] B. Tanner and R. S. Sutton. Td networks: temporal-difference networks with eligibility traces. In *22nd international conference on Machine learning*, pages 888–895, 2005.

[Varakantham *et al.*, 2005] P. Varakantham, R. T. Maheswaran, and M. Tambe. Exploiting belief bounds: practical pomdps for personal assistant agents. In *International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS-05)*, pages 978–985, 2005.

[Wang and Mahadeavan, 1999] G. Wang and S. Mahadeavan. Hierarchical optimization of policy-coupled semi-markov decision processes. In *International Conference on Machine Learning (ICML-99)*, pages 466–473, 1999.

[Wang *et al.*, 2006] T. Wang, P. Poupart, M. Bowling, and D. Schuurmans. Compact convex upper bound iteration for approximate pomdp planning. In *The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference (AAAI-06)*, 2006.

[White, 1991] C. C. White. Partially observable markov decision processes: A survey. *Annals of Operations Research*, 32:215–230, 1991.

[Wiewiora, 2005] E. Wiewiora. Learning predictive representations from a history. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 964–971, 2005.

[Williams *et al.*, 2005] J. D. Williams, P. Poupart, and S. Young. Partially observable markov decision processes with continuous observations for dialogue management. In *The 6th SigDial Workshop on Discourse and Dialogue, Lisbon, Portugal*, 2005.

[Wingate and Singh, 2006] D. Wingate and S. Singh. Kernel predictive linear gaussian models for nonlinear stochastic dynamical systems. In *The 23rd International Conference on Machine learning (ICML-06)*, pages 1017–1024, New York, NY, USA, 2006. ACM Press.

[Wolfe and Singh, 2006] B. Wolfe and S. Singh. Predictive state representations with options. In *International Conference on Machine Learning (ICML-06)*, pages 1025–1032, 2006.

[Wolfe *et al.*, 2005] B. Wolfe, M. R. James, and S. Singh. Learning predictive state representations in dynamical systems without reset. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 980–987, 2005.

[Yu and Bertsekas, 2004] H. Yu and D. Bertsekas. Discretized approximations for pomdp with average cost. In *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI-04)*, page 619, 2004.

[Zhang and Liu, 1996] N. L. Zhang and W. Liu. Planning in stochastic domains: Problem characteristics and approximation. Technical report, Hong Kong University of Technology, 1996.

[Zhang and Zhang, 2001a] N. L. Zhang and W. Zhang. Space-progressive value iteration: An anytime algorithm for a class of pomdps. In *Sixth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, 2001.

[Zhang and Zhang, 2001b] N. L. Zhang and W. Zhang. Speeding up the convergence of value iteration in partially observable markov decision processes. *Artificial Intelligence Research*, 14:29–51, 2001.

[Zhang and Zhang, 2001c] N. L. Zhang and W. Zhang. *Speeding up the convergence of value iteration in partially observable Markov decision processes.* Journal of Artificial Intelligence Research, 2001.

[Zhang *et al.*, 2001] B. Zhang, Q. Cai, J. Mao, and B. Guo. Planning and acting under uncertainty: A new model for spoken dialogue system. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 572–579. Morgan Kaufmann Publishers Inc., 2001.

Document Log:

Manuscript Version 0

Typeset by $\mathcal{AMS}$-LaTeX — 31 January 2007

MASOUMEH TABAEH IZADI

MCGILL UNIVERSITY, 3480 UNIVERSITY ST., MONTRÉAL (QUÉBEC) H3A 2A7, CANADA, *Tel.* : (514) 398-7071

E-mail address: mtabae@cs.mcgill.ca

Typeset by $\mathcal{AMS}$-LaTeX